N° d'ordre : **2011-14-TH**

THÈSE DE DOCTORAT

SPECIALITE : PHYSIQUE

Ecole Doctorale « Sciences et Technologies de l'Information des Télécommunications et des Systèmes »

Présentée par :

Alaa DIB

Commande non linéaire et asservissement visuel de robots autonomes

Soutenue le 21 Octobre 2011 devant les membres du jury :

MME	Yasmina BESTAOUI	Université d'Evry	Examinatrice
M.	Patrick BOUCHER	Supélec	Directeur de thèse
M.	Nacer M'SIRDI	LSIS - CNRS	Rapporteur
MME	Sylvie NAUDET	CEA/LIST	Examinatrice
M.	Roméo ORTEGA	LSS - CNRS	Président
M.	Éric OSTERTAG	ENSPS	Rapporteur
MME	Houria SIGUERDIDJANE	Supélec	Co-Directeur de thèse

To my beloved wife Lama,
and to my precious daughter Laura Anne.

Remerciements

Je souhaite remercier les membres du jury de m'honorer de leur présence lors de cette soutenance de thèse. Merci à Monsieur Romeo Ortega, Directeur de recherche au CNRS, qui m'a fait l'honneur de présider le jury. Merci à Monsieur Nacer M'sirdi, Professeur à LSIS, et à Monsieur Éric Ostertag, Professeur émérite à ENSPS, qui ont accepté d'être rapporteurs pour cette thèse. Je les remercie pour leurs remarques et commentaires qui m'ont permis d'améliorer ce manuscrit de thèse. Je tiens également à remercier Madame Yasmina Bestaoui, Maître de conférences à l'Université d'Evry, et Madame Sylvie Naudet, Ingénieure de recherche au CEA, d'avoir bien voulu participer à mon jury de thèse.

Je remercie mon directeur de thèse Monsieur Patrick Boucher, Chef de Département Automatique de Supélec, pour son accueil chaleureux. Je le remercie également pour sa confiance et ses conseils précieux.

J'exprime toute ma gratitude à mon encadrante Madame Houria Siguerdidjane, Professeur à Supélec, ses conseils m'ont été d'une aide précieuse tout au long de cette thèse. Je la remercie pour la liberté qu'elle m'a laissée dans mes recherches et pour tout le temps qu'elle m'a consacré.

Cette thèse n'aurait pas pu se dérouler dans d'aussi bonnes conditions sans l'ensemble du personnel du département Automatique. Je remercie Monsieur Martial Demerlé pour son assistance technique et nos discussions fructueuses quand j'ai commencé à travailler sur le robot. Je remercie Monsieur Emmanuel Godoy qui était prêt à fournir tout l'équipement dont j'avais besoin dans ma recherche. Je remercie Monsieur Léon Marquet pour son aide technique en cas de besoin. Un grand merci à Madame Josiane Dartron, secrétaire du département Automatique, pour sa gentillesse et sa disponibilité tout au long de cette thèse.

Je tiens à remercier particulièrement les autres doctorants et stagiaires du Département Automatique de Supélec pour leur soutien quotidien et tous les moments partagés durant ces années. Merci à Haitham, Ali, Raluca, Bogdan, Nam, Christina, Nikola, Rayen, Guillermo, Younane, ainsi qu'à ceux que j'ai connus plus brièvement.

Merci enfin à mon épouse Lama d'avoir supporté cette thèse et le temps que j'ai consacré à celle-ci durant ces années. C'est sa présence et son soutien indéfectible qui m'ont permis d'avancer au quotidien et de finir cette thèse. Merci infiniment.

Abstract

This thesis focuses on the problem of moving and localizing an autonomous mobile robot in its local environments. The first part of the manuscript concerns two basic motion tasks, namely the stabilization and trajectory tracking. Two control strategies were discussed: the integral sliding mode, and the method known as "Immersion and Invariance" for nonlinear control. The second part focuses on both 2D and 3D visual servoing techniques. Image moments were chosen as visual features as they provide a more geometric and intuitive meaning than other features, and they are less sensitive to image noise and other measurement errors. A new approach to visual servoing based on image is herein proposed. It is based on the generation of trajectories directly on the image plane (Calculation of the image features corresponding to a given Cartesian path). This approach ensures that the robustness and stability are extended due to the fact that the initial and desired locations of the camera are close. The trajectories obtained guarantee that the target remains in the field of view of the camera and the corresponding movement of the robot is physically feasible. Experimental tests have been conducted, and satisfactory results have been obtained from both implementations regarding motion control and visual servoing strategies. Although developed and tested in the specific context of a unicycle type robot, this work is generic enough to be applied to other types of vehicles.

Contents

Résumé

Introduction

Par robot on entend un système mécanique équipé de capteurs, d'actionneurs et d'un système de commande permettant d'agir sur les actionneurs en fonction d'une part de la tâche à accomplir, et d'autre part, des informations données par les capteurs. Actuellement, de tels systèmes de commande sont bien évidemment des ordinateurs. Un robot est dit autonome si, moyennant une spécification externe 'de haut niveau' de la tâche à accomplir, le robot est capable de la mener à bien sans intervention humaine. L'utilité de robots autonomes ou même partiellement autonomes est incontestable et les domaines d'application sont multiples et variés. Les domaines les plus attractifs dans le futur concernant la robotique de service, la robotique médicale, la robotique environnementale par exemple. Néanmoins, le développement d'un robot autonome pose encore de nombreux problèmes fondamentaux : le problème de la perception de l'environnement par exemple est naturellement lié au domaine de la vision, le problème du suivi de trajectoire est lié à la théorie de la commande. Un autre problème fondamental est celui de la planification de trajectoires.

L'objectif de cette thèse est de concevoir et de mettre en œuvre des lois d'asservissement permettant à un robot mobile d'exécuter une tâche ou de réaliser une mission donnée. Pour cela, il est nécessaire de déterminer les mouvements de déplacement du robot par rapport à son environnement local. Les mouvements sont réalisés à l'aide de la planification de trajectoires, l'évitement d'obstacle, la commande de mouvement, la localisation, la perception et la navigation. Dans le cadre de ce travail, on s'intéresse alors à des aspects méthodologiques :

- Une planification réactive de trajectoire afin de trouver une trajectoire sans collision à partir d'une position initiale donnée à un point cible prédéfini en présence

d'obstacles fixes.

- Proposer des stratégies de commande permettant à un robot mobile d'effectuer une tâche donnée d'une manière indépendante, c'est-à-dire uniquement à partir des informations fournies en temps réel par des capteurs.
- Une étape très importante de l'autonomie avancée est l'asservissement visuel. Nous allons donc nous attacher au problème de commande et à la réalisation des tâches basées sur la vision pour la navigation de robot mobile et l'exécution des mouvements complexes.
- Effectuer des simulations et des tests expérimentaux sur un robot mobile pour valider les stratégies proposées.

Organisation du document

Ce document est composé de cinq chapitres.

Au **chapitre 1**, nous présentons la modélisation du robot mobile de type unicycle, nous commençons par décrire la position du robot dans l'espace de configuration. La structure des contraintes découlant de la cinématique de roulement des roues est ensuite analysés. Il est montré que ces contraintes sont en général non holonomes et par conséquence réduisent la mobilité locale du robot. Enfin, le modèle dynamique du robot mobile est présenté à la fin de ce chapitre.

Le **chapitre 2** est dédié au problème de planification de trajectoire en présence de contraintes non holonomes. L'existence de sorties plates est exploitée pour concevoir des méthodes de planification de trajectoires qui garantissent que les contraintes non holonomes sont satisfaites, et enfin, la planification de trajectoires en présence d'obstacles est présentée.

Au **chapitre 3**, nous abordons le problème de stratégies de commande, en référence aux deux tâches de mouvement de base, à savoir, la stabilisation et le suivi de trajectoire. Tout d'abord, nous discutons la linéarisation par retour d'état combiné avec un contrôleur robuste basé sur les modes glissants intégraux. Ensuite, nous étudions la méthode d'*Immersion et Invariance* pour évaluer son degré d'applicabilité et sa performance par rapport à des méthodes plus classiques. Des simulations sont effectuées pour montrer les performances des deux techniques de commande.

Le **chapitre 4** traite de l'asservissement visuel. Nous commencons d'abord par présenter la modélisation du système (caméra + robot), puis nous présentons quelques algorithmes de base pour le traitement de l'image, visant à extraire de l'information numérique dénommée primitives d'image (principalement moments de l'image). Ces paramètres, par rapport aux images d'objets présents dans la scène observée par une caméra, peuvent être utilisés pour estimer la pose de la caméra par rapport aux objets et vice versa. À cette fin, les méthodes analytiques d'estimation de pose sont présentées. Une opération fondamentale pour l'asservissement visuel est le calibrage de la caméra ; à cette fin, une méthode de calibrage basée sur la mesure d'un certain nombre de correspondances est présentée. Ensuite, les deux approches principales de l'asservissement visuel sont introduites, à savoir l'asservissement visuel basé sur la position et l'asservissement visuel basé sur l'image. L'utilisation des moments d'image en tant que primitives visuelles dans les deux approches est également présentée. L'idée novatrice ici, à notre connaissance au moins, consiste à effectuer l'asservissement visuel 2D en utilisant les moments d'images comme indices visuels et en planifiant la trajectoire dans le plan d'image, au lieu de la planifier dans l'espace cartésien du robot comme ce qui est utilisé de coutume dans la littérature.

Le **chapitre 5** est enfin consacré aux résultats expérimentaux sur la commande de mouvement et les techniques d'asservissement visuel sur deux robots mobiles : KOALA et PEKEE II, afin d'évaluer la qualité et l'applicabilité de ces techniques. Les tests expérimentaux ont montré la validité de notre démarche.

Chapitre 1 : Modélisation

Le robot mobile de type unicycle est largement utilisé dans la litterature (Fig. 1). Ce type de robot est surtout utilisé pour des applications intérieures. Son mécanisme d'entraînement a deux moteurs indépendants, chacun d'eux entraîne une roue du robot.

Son modèle cinématique est

$$\dot{\mathbf{q}} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \mathbf{v} = \mathbf{g}_1(\mathbf{q})v + \mathbf{g}_2(\mathbf{q})\omega \tag{1}$$

Le vecteur $\mathbf{q} = [x \quad y \quad \theta]^T$ représente la position linéaire et angulaire du robot, le vecteur $\mathbf{v} = [v \quad \omega]^T$ représente les vitesses linéaires et angulaires et $\mathbf{g}_1$, $\mathbf{g}_2$ sont les deux

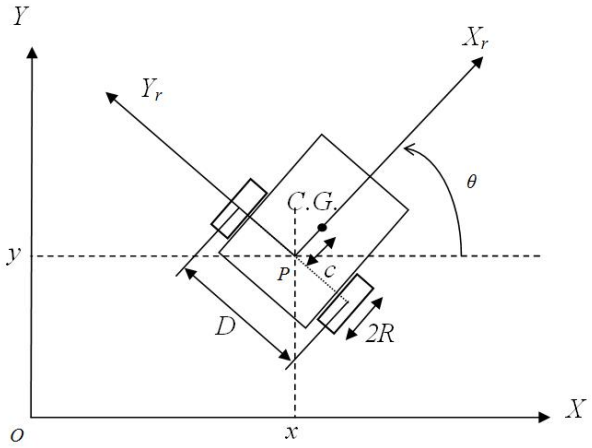


Figure 1: Robot mobile de type unicycle.

champs de vecteurs de commande.

Le modèle cinématique de l'unicycle a les propriétés suivantes :

- L'existence d'une contrainte non holonome :

$$\dot{x}\sin\theta - \dot{y}\cos\theta = [\sin\theta \quad -\cos\theta \quad 0]\,\dot{\mathbf{q}} = 0 \tag{2}$$

- Commandabilité : Afin d'étudier la commandabilité de l'unicycle, nous utilisons des outils de la théorie de la commande non linéaire [52]. Puisque le système est 'sans dérive', la commandabilité peut être vérifiée à l'aide de la condition d'accessibilité.

$$\text{rang}([\mathbf{g}_1 \quad \mathbf{g}_2 \quad [\mathbf{g}_1, \mathbf{g}_2]]) = \text{rang}(\begin{bmatrix} \cos\theta & 0 & \sin\theta \\ \sin\theta & 0 & -\cos\theta \\ 0 & 1 & 0 \end{bmatrix}) = 3 = n \tag{3}$$

où $[\mathbf{g}_1, \mathbf{g}_2]$ est le crochet de Lie des deux champs de vecteurs de commande, la condition (3) implique la commandabilité du système.

- Commandabilité par rapport à une trajectoire : Pour que le robot effectue un mouvement cartésien souhaité, il est plus commode de générer une trajectoire d'état correspondante $\mathbf{q}_d(t) = [x_d(t) \;\; y_d(t) \;\; \theta_d(t)]^T$. Afin que cette trajectoire soit

réalisable, elle doit satisfaire la contrainte non holonome sur le déplacement du véhicule, en d'autres termes, elle doit satisfaire les équations

$$\begin{aligned} \dot{x}_d &= v_d \cos\theta_d \\ \dot{y}_d &= v_d \sin\theta_d \\ \dot{\theta}_d &= \omega_d \end{aligned} \tag{4}$$

pour un choix de v_d et ω_d. Si on définit les erreurs de suivi suivantes

$$\mathbf{e} = \begin{bmatrix} \mathbf{e}_1 \\ \mathbf{e}_2 \\ \mathbf{e}_3 \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_d - x \\ y_d - y \\ \theta_d - \theta \end{bmatrix} \tag{5}$$

et en utilisant la transformation

$$v = v_d \cos e_3 - u_1 \tag{6}$$

$$\omega = \omega_d - u_2 \tag{7}$$

puis en linéarisant et en calculant la matrice de commandabilité du système linéarisé, on obtient

$$\mathcal{C} = \begin{bmatrix} B & AB & A^2B \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & -\omega_d^2 & v_d\omega_d \\ 0 & 0 & -\omega_d & v_d & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{8}$$

Ce système est commandable, étant donné que $v_d \neq 0$ ou $\omega_d \neq 0$.

- Sorties plates : en choisissant le vecteur de sortie $(x(t), y(t))$, on peut écrire

$$\begin{aligned} \theta(t) =& \text{atan2}(\dot{y}(t), \dot{x}(t)) \\ v(t) =& \pm\sqrt{(\dot{x}(t))^2 + (\dot{y}(t))^2} \\ \omega(t) =& \frac{\dot{x}(t)\ddot{y}(t) - \ddot{x}(t)\dot{y}(t)}{(\dot{x}(t))^2 + (\dot{y}(t))^2} \end{aligned} \tag{9}$$

Le vecteur d'état $\mathbf{x}$ et le vecteur d'entrées $\mathbf{u}$ qui peuvent alors s'exprimer en fonction de $\mathbf{y}$ et de ses dérivées, le système est donc plat.

Reprenons le modèle cinématique (1) de l'unicycle. Avec le changement de coordonnées

$$\begin{aligned} z_1 &= \theta \\ z_2 &= x\cos\theta + y\sin\theta \\ z_3 &= x\sin\theta - y\cos\theta \end{aligned} \tag{10}$$

et la transformation

$$\begin{aligned} v &= u_2 + z_3 u_1 \\ \omega &= u_1 \end{aligned} \tag{11}$$

on obtient la forme en chaîne

$$\begin{aligned} \dot{z}_1 &= u_1 \\ \dot{z}_2 &= u_2 \\ \dot{z}_3 &= z_2 u_1 \end{aligned} \tag{12}$$

Le modèle dynamique est obtenu en utilisant la méthode de Newton-Euler pour trouver les accélérations des centres de masse. Ici, le robot se compose de trois corps rigides : le corps du robot et deux roues. Cela nous permet d'obtenir le modèle dynamique du robot

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \omega \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{1}{mR} & \frac{1}{mR} \\ \frac{D}{2IR} & -\frac{D}{2IR} \end{bmatrix} \begin{bmatrix} \tau_r \\ \tau_l \end{bmatrix} \tag{13}$$

où

$$\begin{aligned} m &= m_r + 2\left(m_w + \frac{I_{wy}}{R^2}\right) \\ I &= I_r + 2I_w + \left(m_w + \frac{I_{wy}}{R^2}\right)\frac{D^2}{2} \end{aligned} \tag{14}$$

τ_r et τ_l sont les couples appliqués sur les roues droite et gauche respectivement, m_r est la masse du robot, m_w est la masse de la roue, I_r est le moment d'inertie du robot autour de son axe vertical, I_w est le moment d'inertie de la roue autour de son axe vertical, I_{wy} est le moment d'inertie de la roue autour de son axe horizontal, R est le rayon des roues et D est la distance entre leurs centres.

En utilisant la transformation

$$\tau = \begin{bmatrix} \frac{mR}{2} & \frac{IR}{D} \\ \frac{mR}{2} & -\frac{IR}{D} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{15}$$

on obtient le modèle qu'on appelle le modèle cinématique du deuxième ordre

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \omega \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{16}$$

Une étape très importante pour l'autonomie du robot mobile est la planification de trajectoire, qu'on va discuter dans le prochain chapitre.

Chapitre 2 : Planification de trajectoires

Le problème de la planification d'une trajectoire pour un robot mobile peut être divisé en deux sous-problèmes : trouver un chemin et définir une loi temporelle sur le chemin. Toutefois, si le robot mobile est soumis à des contraintes non holonomes, le premièr sous-problème devient plus difficile. En fait, en plus de satisfaire les conditions aux limites (interpolation des points attribués et continuité du degré désiré) le chemin doit également satisfaire les contraintes non holonomes.

2.1 Planification de chemin

Les sorties plates de l'unicycle permettent de résoudre le problème de planification plus efficacement. Prenons le problème de conduire un unicycle à partir d'une configuration initiale $\mathbf{q}(s_i) = \mathbf{q}_i = [x_i \;\; y_i \;\; \theta_i]^T$ à une configuration finale $\mathbf{q}(s_f) = \mathbf{q}_f = [x_f \;\; y_f \;\; \theta_f]^T$.

Le problème de la planification de chemin peut être résolu en utilisant les polynômes cubiques. Un polynôme cubique a 4 coefficients, et par conséquent, peut être utilisé pour satisfaire à la fois les contraintes de position et de vitesse à la position initiale et finale. Par interpolation des valeurs initiales x_i, y_i et des valeurs finales x_f, y_f des sorties plates x, y, et en posant $s_i = 0$ et $s_f = 1$, on peut utiliser les polynômes cubiques suivants

$$\begin{aligned} x(s) &= s^3 x_f - (s-1)^3 x_i + \alpha_x s^2(s-1) + \beta_x (s-1)^2 \\ y(s) &= s^3 y_f - (s-1)^3 y_i + \alpha_y s^2(s-1) + \beta_y (s-1)^2 \end{aligned} \tag{17}$$

avec

$$\begin{aligned} \alpha_x &= k\cos\theta_f - 3x_f \quad & \alpha_y &= k\sin\theta_f - 3y_f \\ \beta_x &= k\cos\theta_i + 3x_i \quad & \beta_y &= k\sin\theta_i + 3y_i \end{aligned}$$

Si le système est exprimé sous forme de chaîne, dans ce cas, sous l'hypothèse $z_{1,i} \neq z_{1,f}$, on a le chemin avec des polynômes cubiques

$$\begin{aligned} z_1(s) &= z_{1,f}\, s - (s-1) z_{1,i} \\ z_3(s) &= s^3 z_{3,f} - (s-1)^3 z_{3,i} + \alpha_3 s^2(s-1) + \beta_3 s (s-1)^2 \end{aligned} \tag{18}$$

avec

$$\alpha_3 = z_{2,f}(z_{1,f} - z_{1,i}) - 3z_{3,f}$$
$$\beta_3 = z_{2,i}(z_{1,f} - z_{1,i}) + 3z_{3,i}$$

2.2 Planification de chemin en présence d'obstacles

À cette fin, nous avons appliqué l'algorithme *TangentBug* [56], que nous avons modifié pour le rendre plus adéquat à l'exigence de commande.

2.3 Planification de trajectoires

La planification de trajectoire consiste à choisir une loi temporelle $s = s(t)$ pour un certain chemin $\mathbf{q}(s), s \in [s_i, s_f]$ en respectant les limitations des actionneurs

$$\begin{aligned} |v(t)| &\leq v_{max}, \qquad \forall t \\ |\omega(t)| &\leq \omega_{max}, \qquad \forall t \end{aligned} \tag{19}$$

Il est possible de ralentir la loi temporelle par changement d'échelle uniforme dans le cas où les vitesses sont inadmissibles. Pour cela, il est commode de réécrire la loi temporelle par la substitution de t par la variable temporelle normalisée $\tau = t/T$, avec $T = t_f - t_i$.

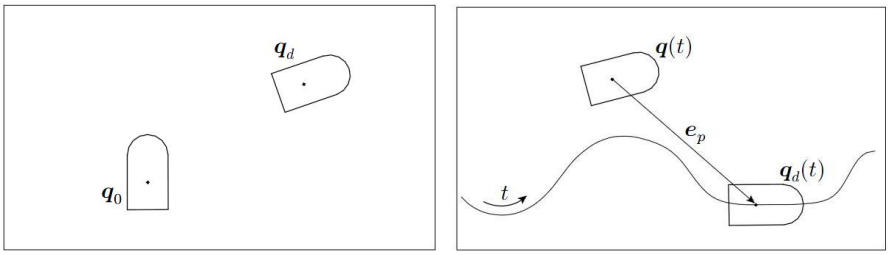
Une fois qu'une trajectoire a été planifiée, une stratégie de commande devrait être mise en œuvre pour suivre cette trajectoire ou pour réaliser une posture donnée.

Chapitre 3 : Différentes stratégies de commande

Dans le cas des robots mobiles à roues, il est possible de distinguer deux problèmes de commande (Fig. 2) :

- *Stabilisation* : le robot doit atteindre asymptotiquement une configuration désirée $\mathbf{q}_d$, à partir d'une configuration initiale $\mathbf{q}_0$.
- *Suivi de trajectoires* : le robot doit suivre asymptotiquement une trajectoire cartésienne désirée $(x_d(t), y_d(t))$, à partir d'une configuration initiale $q_0 = [x_0 \ y_0 \ \theta_0]^T$ qui peut ou pas être sur la trajectoire.

Dans ce chapitre, nous discutons de deux stratégies de commande : une stratégie basée sur les *modes glissants intégraux* combinés par linéarisation par retour d'état, et l'autre est basé sur la méthode d'*Immersion et Invariance*.



(a) Stabilisation **(b)** Suivi de trajectoires

Figure 2: Commande de mouvement de l'unicycle

3.1 Linéarisation par bouclage

Considérons le système non-holonome

$$\begin{aligned} \dot{\mathbf{q}} &= \mathbf{G}(\mathbf{q})\mathbf{v} \\ \dot{\mathbf{v}} &= \mathbf{u} \end{aligned} \tag{20}$$

où $\mathbf{q} \in \mathbb{R}^n$ et $\mathbf{v} \in \mathbb{R}^m$, et qui est soumis à $n - m$ contraintes non holonomes.

En choisissant la fonction de sortie

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} \tag{21}$$

La linéarisation par bouclage est réalisée par la commande suivante

$$\mathbf{u} = \mathbf{D}^{-1}(\mathbf{q})(\mathbf{z} - \mathbf{F}(\mathbf{q}, \mathbf{v})) \tag{22}$$

où

$$\mathbf{F}(\mathbf{q}, \mathbf{v}) = \frac{\partial}{\partial \mathbf{q}} \left[\nabla_{\mathbf{q}} h(\mathbf{q})\mathbf{G}(\mathbf{q})\mathbf{v}\right] \mathbf{G}(\mathbf{q})\mathbf{v} \tag{23}$$

$$\mathbf{D}(\mathbf{q}) = \nabla_{\mathbf{q}} h(\mathbf{q})\mathbf{G}(\mathbf{q}) \tag{24}$$

Dans le cas de l'unicycle, on choisit la fonction de sortie suivante

$$\mathbf{y} = h(\mathbf{q}) = \begin{bmatrix} x + L\cos\theta \\ y + L\sin\theta \end{bmatrix} \tag{25}$$

avec $L \neq 0$, d'où

$$\mathbf{D}(\mathbf{q}) = \begin{bmatrix} \cos\theta & -L\sin\theta \\ \sin\theta & L\cos\theta \end{bmatrix} \tag{26}$$

et

$$\mathbf{F}(\mathbf{q}) = \begin{bmatrix} -v\omega\sin\theta - L\omega^2\cos\theta \\ v\omega\cos\theta - L\omega^2\sin\theta \end{bmatrix} \tag{27}$$

3.2 Commande par mode de glissement intégral

L'avantage de la commande par mode de glissement est sa robustesse vis à vis des perturbations et des incertitudes structurelles, c.-à-d. la réponse du système est peu sensible aux variations des paramètres du système et des perturbations externes. Toutefois, pendant la phase d'accrochage (avant que le glissement se produise), le système n'a pas une telle insensibilité, par conséquent, elle ne peut être assurée à travers la réponse complète. La robustesse au cours de la phase d'accrochage est normalement améliorée par commande à gain élevé. Les problèmes de stabilité qui se posent limitent inévitablement ses applications.

La notion de mode de glissement intégral se concentre sur la robustesse de la commande dans l'espace d'état tout entier. L'ordre de l'équation du mouvement dans ce type de mode de glissement est égal à la dimension de l'espace d'état. Par conséquent, la robustesse du système peut être garantie tout au long d'une réponse complète du système à partir du temps initial.

Pour un système dynamique

$$\dot{x} = f(x) + B(x)u \tag{28}$$

où $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$, on suppose qu'il existe une loi de commande $u = u_0(x)$, telle que le système (28) peut être stabilisé :

$$\dot{x}_* = f(x) + B(x)u_0 \tag{29}$$

Le système (28) est normalement sous certaines conditions d'incertitude qui peuvent être générées par les variations des paramètres, modes non modélisés et perturbations externes, etc. Donc, le système en réalité peut être donné par

$$\dot{x} = f(x) + B(x)u + h(x,t) = f(x) + B(x)u + B(x)u_h \quad u_h \in \mathbb{R}^m \tag{30}$$

Tout d'abord, on conçoit une loi decommande

$$u = u_0 + u_1 \tag{31}$$

où u_1 est une commande discontinue permettant de rejeter la perturbation $h(x,t)$. Ensuite, on conçoit *la fonction de commutation s*

$$s = s_0(x) + \mu \tag{32}$$

avec $s, s_0(x), \mu \in \mathbb{R}^m$.

La strategie de commande consiste à faire en sorte que la commande équivalente satisfasse

$$u_{1eq} = -u_h \tag{33}$$

La condition (33) est satisfaite si on choisit

$$\begin{aligned} \dot{\mu} &= \frac{\partial s_0}{\partial x}\left(f(x) + B(x)u_0\right) \\ \mu(0) &= -s_0(x(0)) \end{aligned} \tag{34}$$

La commande par mode de glissement est alors

$$u_1 = -M(x)sign(s) \tag{35}$$

3.2.1 Stabilisation

Après la linéarisation du système (20), le système en boucle fermée est équivalent à deux sous-systèmes linéaires de la forme

$$\dot{\mathbf{x}}_i = \mathbf{B}_{ii} z_i \qquad i = 1, 2 \tag{36}$$

Et la commande calculée en utilisant la méthode des modes glissants intégraux est donnée par

$$\begin{aligned} z_{i0} &= -\mathbf{k}^T \mathbf{x}_i + m_0(\mathbf{x})sign(s_i) \\ s_i &= \mathbf{c}_i^T \mathbf{x}_i + \mu_i \\ \dot{\mu}_i &= -\mathbf{c}_i^T(\mathbf{B}_{ii} z_{i0}) \\ \mu_i(0) &= -\mathbf{c}_i^T \mathbf{x}_i(0) \end{aligned} \tag{37}$$

Le vecteur de gain $\mathbf{k}$ peut être déterminé par les méthodes de placement de pôles ou par optimisiation quadratique, de type Régulateur Linéaire Quadratique (LQR).

3.2.2 Suivi de trajectoires

Étant donné une trajectoire de référence continue

$$\mathbf{y}_d(t) = h(\mathbf{q}_d(t)) \tag{38}$$

le problème de suivi revient à concevoir une loi de commande pour le système (20) avec la sortie $\mathbf{y}(t) = h(\mathbf{q}(t))$ telles que l'erreur de suivi

$$\mathbf{e}(t) = \mathbf{y}(t) - \mathbf{y}_d(t) \tag{39}$$

soit bornée et tende asymptotiquement vers zéro. La commande est donc donnée par

$$\begin{aligned} z_{i0} &= -\mathbf{k}^T\mathbf{x}_i + m_0(\mathbf{x})sign(s_i) + \ddot{\mathbf{y}}_d \\ s_i &= \mathbf{c}_i^T\mathbf{x}_i + \mu_i \\ \dot{\mu}_i &= -\mathbf{c}_i^T(\mathbf{B}_{ii}z_{i0}) \\ \mu_i(0) &= -\mathbf{c}_i^T\mathbf{x}_i(0) \end{aligned} \tag{40}$$

3.3 Immersion et invariance (I&I)

La notion d'invariance joue un rôle fondamental dans la théorie de la commande des systèmes non linéaires. Les notions de distribution et de variétés invariantes ont permis d'analyser la stabilité des systèmes non linéaires pour donner lieu au développement de nouvelles lois de commande, voir par exemple [52]. Un autre concept essentiel est l'immersion. Il s'agit de transformer le système en un autre système de dimension potentiellement supérieure, qui conserve les propriétés du premier. Plus précisément, l'immersion d'un système est une application vers un espace de dimension plus élevée.

Récemment, dans [7], les notions géométriques de variétés invariantes et d'immersion d'un système ont été exploitées pour fournir un cadre nouveau pour la construction de lois de commande non linéaires sans avoir recours à la théorie de Lyapunov. Ainsi, l'approche I&I propose de capturer le comportement désiré du système à commander à l'aide d'un système cible auxiliaire. Le problème de commande se réduit alors à la conception d'une loi de commande qui garantisse que le système commandé

$$\dot{x} = f(x) + g(x)\,u\ , \quad x \in \mathbb{R}^n, \quad u \in \mathbb{R}^m \tag{41}$$

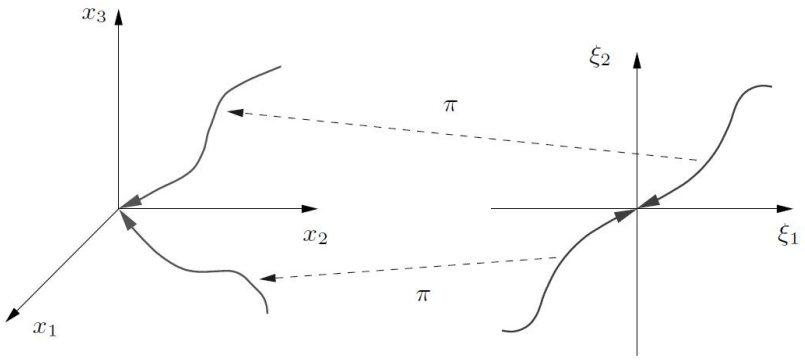


Figure 3: Représentation graphique de l'approche par immersion et invariance.

converge asymptotiquement vers le système cible, avec un point d'équilibre (globalement) asymptotiquement stable et décrit par

$$\dot{\xi} = \alpha(\xi), \quad \xi \in \mathbb{R}^p, \quad p < n \tag{42}$$

et une application

$$x = \pi(\xi) \tag{43}$$

telle que

$$f\left(\pi(\xi)\right) + g\left(\pi(\xi)\right) c\left(\pi(\xi)\right) = \frac{\partial \pi}{\partial \xi} \alpha(\xi) \tag{44}$$

pour une certaine application $c \in \mathbb{R}^n$. Les propriétés de stabilité du système cible sous la condition (43) impliquent que le point d'équilibre désiré peut-être stabilisé asymptotiquement, tandis que l'équation différentielle partielle (44) implique que la variété $x - \pi(\xi) = 0$ est invariante vis-à-vis des dynamiques du système étendu (41)-(42). La convergence vers la variété est assurée par la condition : Toutes les trajectories du système

$$\dot{z} = \frac{\partial \phi}{\partial x} \left(f(x) + g(x)\psi(x, z)\right) \tag{45}$$

$$\dot{x} = f(x) + g(x)\psi\left(x, z\right) \tag{46}$$

sont bornées et satisfont

$$\lim_{t\to\infty} z(t) = 0 \tag{47}$$

Le cœur du problème consiste donc à trouver une variété qui peut être rendue invariante et attractive, avec la dynamique interne comme une copie de la dynamique en boucle fermée désirée, et à concevoir une loi de commande qui oriente l'état du système suffisamment proche de cette variété.

3.3.1 Stabilisation

Considérons le système sous forme (10), en choisissant la dynamique cible

$$\Sigma_T : \left\{ \begin{array}{rcll} \dot{\xi}_1 & = & -\alpha_1\xi_1, & \alpha_1 > 0 \\ \dot{\xi}_2 & = & -\alpha_2\xi_2, & \alpha_2 > 0 \end{array} \right. \tag{48}$$

et l'application

$$\pi(\xi) = \left[\begin{array}{c} \xi_1 \\ \xi_2 \\ \frac{b}{b+1}\xi_1\xi_2 \end{array} \right], \qquad b = \alpha_1/\alpha_2 \tag{49}$$

on obtient la loi de commande en utilisant l'équation (44), et en assurant la condition (45)

$$\left\{ \begin{array}{l} u_1 = \lambda_1 x_1 + \lambda_2 \dfrac{x_3}{x_2} \\ u_2 = -\alpha_2 x_2 \end{array} \right. \tag{50}$$

où $\lambda_1 = b(\gamma - \alpha_2)$, $\lambda_2 = -\gamma(b+1)$ et $\gamma > 0$.

3.3.2 Suivi de trajectoires

Considérons la dynamique de l'erreur de suivi d'une trajectroire décrite par les équations (4)

$$\left[\begin{array}{c} \dot{e}_1 \\ \dot{e}_2 \\ \dot{e}_3 \end{array} \right] = \left[\begin{array}{c} \omega_d e_2 \\ v_d \sin e_3 - \omega_d e_1 \\ 0 \end{array} \right] + \left[\begin{array}{cc} 1 & -e_2 \\ 0 & e_1 \\ 0 & 1 \end{array} \right] \left[\begin{array}{c} \eta_1 \\ \eta_2 \end{array} \right] \tag{51}$$

où

$$\begin{array}{rcl} \eta_1 & = & v_d \cos e_3 - v \\ \eta_2 & = & \omega_d - \omega \end{array}$$

en choisissant la dynamique cible

$$\Sigma_T : \left\{ \dot{\xi} = -v_d \sin \operatorname{atan}(r\xi) \, , \quad r > 0 \right. \tag{52}$$

et l'application

$$\pi(\xi) = \begin{bmatrix} 0 \\ \xi \\ \pi_3(\xi) \end{bmatrix} \tag{53}$$

on obtient la loi de commande en utilisant les équations (44), et en assurant la condition (45)

$$\begin{aligned} \eta_1 (\mathbf{e}, z) =& - k_1 e_1 \\ \eta_2 (\mathbf{e}, z) =& \frac{1 + r^2 e_2^2}{1 + r^2 e_2^2 + r e_1} [-\gamma v_d \mu e_2 + \frac{r}{1 + r^2 e_2^2} (-v_d \sin(e_3) + w_d e_1) - k_2 \gamma z_2] \end{aligned} \tag{54}$$

où $k_1, k_2, \gamma > 0$, $z_2 = e_3 + \operatorname{atan}(r e_2)$ et

$$\mu(e_2, z_2) = \frac{1}{\sqrt{1 + r^2 e_2^2}} \frac{\sin z_2}{z_2} + \frac{r e_2}{\sqrt{1 + r^2 e_2^2}} \cdot \frac{1 - \cos z_2}{z_2} \tag{55}$$

La commande de mouvement d'un système robotique dépend des informations fournies par les capteurs. Parmi ceux-ci, les capteurs visuels apportent aux robots une plus grande connaissance de l'environnement dans lequel ils évoluent. Ainsi, il est tout naturel que l'asservissement visuel joue un rôle important dans l'autonomie du robot, point qui est abordé dans le prochain chapitre.

Chapitre 4 : Asservissement Visuel

Les techniques d'asservissement visuel consistent à utiliser les informations fournies par une ou plusieurs caméras afin de commander les mouvements d'un système robotique.

Les systèmes de commande basés sur la vision peuvent être divisés en deux catégories : ceux qui réalisent l'asservissement visuel dans l'espace opérationnel, aussi appelé *asservissement visuel basé sur la position* (3D), et ceux qui réalisent l'asservissement visuel dans l'espace image (2D). La principale différence réside dans le fait que les systèmes de commande de la première catégorie utilisent les mesures visuelles pour reconstruire la pose relative de l'objet par rapport au robot, ou vice versa, alors que les systèmes de la deuxième catégorie sont basés sur la comparaison des paramètres de l'image courants

et désirés. Il existe également des méthodes combinant des caractéristiques communes aux deux catégories, qui peuvent être classées comme *asservissement visuel hybride*.

Toutes les méthodes d'asservissement visuel dépendent de la connaissance des paramètres visuels de l'image (primitives de l'image), ce qui rend l'extraction de l'image une étape clé pour l'asservissement visuel. Une image contient un très grand nombre de données représentées sous forme brute (pixels). Une grande partie de ces données ne comporte pas d'information pertinente. Dans le cadre d'une application en robotique (notamment en temps réel), il est indispensable de pouvoir identifier les fractions de l'image qui véhiculent des informations pertinentes en fonction du type de primitives du modèle que l'on cherche à identifier dans la scène. Il est également nécessaire de représenter ces informations de façon concise dans la mémoire du système.

À cette fin, deux opérations de base sont nécessaires :

- La segmentation d'images : elle se compose d'un processus de regroupement, par lequel l'image est divisée en un certain nombre de groupes, appelés segments, de telle sorte que les composants de chaque groupe soient similaires en ce qui concerne une ou plusieurs caractéristiques. Typiquement, les segments distincts de l'image correspondent à des objets distincts de l'environnement, ou parties d'objets homogènes. Il existe deux approches au problème de la segmentation d'images :
  - segmentation basée sur les régions.
  - segmentation basée sur les contours.
- L'interprétation d'images est le processus de calcul des primitives de l'image, representées par des régions ou par des contours. L'interprétation utilisée dans les applications d'asservissement visuel exige parfois le calcul des *moments d'image*. L'avantage d'utiliser ces moments en asservissement visuel vient du fait qu'ils fournissent une représentation générique d'un objet quelconque, avec des formes simples ou complexes, qui peuvent être segmentées en une image. Ils fournissent également un sens plus géométrique et intuitif que les autres primitives. En outre, le fait que les moments d'image sont calculés sur toute la région les rend moins vulnérables au bruit d'image et aux autres erreurs de mesure.

4.1 Modélisation de la caméra

Le modèle de la caméra est l'ensemble des lois géométriques définissant la façon dont un point de l'espace à trois dimensions se projette sur un plan à deux dimensions, lors du processus de saisie d'une image.

Adoptant la représentation en coordonnées homogènes, un point arbitraire $\tilde{\mathbf{p}}_b = [x \quad y \quad z \quad 1]_b^T$ exprimé dans le repère objet se projette sur le plan image en un pixel $\tilde{\mathbf{p}}_I[x_I \quad y_I \quad 1]$ grâce à la transformation

$$\lambda \tilde{\mathbf{p}}_I = \mathbf{K}\mathbf{T}_b^c \tilde{\mathbf{p}}_b \tag{56}$$

Où $\mathbf{T}_b^c$ représente la matrice de transformation homogène entre le repère de l'objet et le repère de la caméra, $\mathbf{K}$ est la matrice de projection perspective et λ est un facteur d'échelle scalaire.

4.2 Calibrage de la caméra

Une fois le modèle de la caméra choisi, ses paramètres doivent être identifiés. Les valeurs estimées de ces paramètres pour un appareil photo sont obtenues par calibrage. C'est une étape nécessaire pour toute application de vision. De nombreuses techniques ont été développées, et peuvent être classées en deux catégories

- **Calibrage basé sur des objets 3D de référence** : cette technique utilise l'observation d'objets 3D de coordonnées connues. Les objets de calibrage (mire) sont généralement des points répartis sur des plans orthogonaux ou sur un plan translaté dans la direction de sa normale. Le calcul peut alors être effectué de façon relativement simple.

- **Calibrage automatique** : Le mouvement connu de la caméra filmant une scène statique est utilisé pour poser des contraintes sur les paramètres intrinsèques prenant en compte la rigidité des objets filmés en utilisant uniquement les informations de l'image.

4.3 Asservissement visuel basé sur la position (3D)

L'asservissement visuel 3D utilise en entrée de la boucle de commande des informations tridimensionnelles, à savoir la posture $\mathbf{q}$ de la caméra, par rapport à l'objet d'intérêt. La

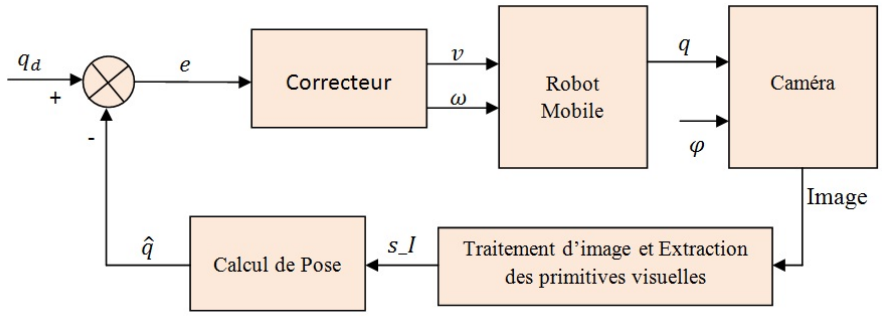


Figure 4: Schéma bloc de l'asservissement visuel basé sur la position.

tâche à réaliser s'exprime alors sous la forme d'une situation de référence $\mathbf{q}_d$ à atteindre. La commande repose ainsi sur la détermination de la situation $\mathbf{q}$ de la caméra, à partir des informations visuelles extraites de l'image, cette opération est appelée *l'estimation de pose*

En choisissant deux points de l'objet cible (x_{o1}, y_{o1}) et (x_{o2}, y_{o2}), leurs coordonnées dans le repère robot (x_{r1}, y_{r1}) et (x_{r2}, y_{r2}), respectivement, peuvent être obtenues en utilisant la formule de projection de la camera (56). Ces coordonnées peuvent être ensuite utilisées pour calculer la position et l'orientation du robot.

À cause du bruit affectant les mesures des coordonnées dans le plan image, les résultats de cette méthode sont affectés par des erreurs. Pour rendre l'estimation de la pose plus robuste, un certain nombre de points $n > 2$ sont considérés et $\mathbf{q}$ est calculée en employant les techniques des moindres carrés. Une fois que $\mathbf{q}$ est déterminé en utilisant des données de l'image, le problème devient un problème de commande de mouvement, et le schéma bloc de l'asservissement visuel 3D est représenté sur le figure 4.

4.4 Asservissement visuel basé sur l'image (2D)

Les techniques d'asservissement visuel 2D utilisent directement les informations visuelles, notées $\mathbf{s}$, extraites de l'image. La tâche à réaliser est alors spécifiée directement dans l'image en termes des primitives visuelles de référence $\mathbf{s}_d$ à atteindre. La loi de commande consiste alors à commander le mouvement de la caméra de manière à annuler l'erreur entre les informations visuelles courantes $\mathbf{s}(t)$ et les primitives désirées $\mathbf{s}_d$. Cette approche permet donc de s'affranchir de l'étape de reconstruction 3D de la cible.

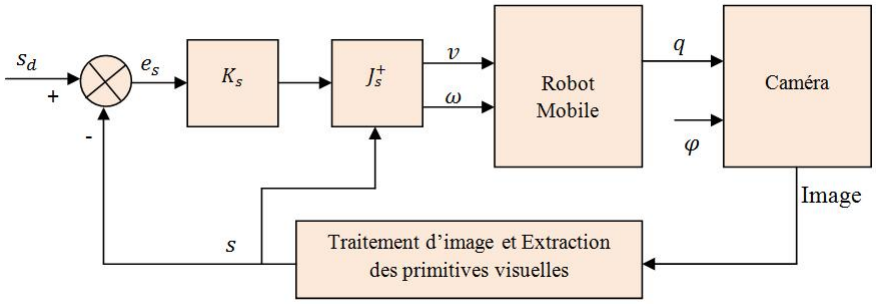


Figure 5: Schéma bloc de l'asservissement visuel basé sur l'image.

Le choix des informations visuelles et l'obtention de la relation les liant au mouvement de la caméra sont deux aspects fondamentaux de l'asservissement visuel 2D. Cette relation, obtenue par dérivation des informations visuelles $\mathbf{s}$ par rapport à la situation $\mathbf{q}$ de la caméra, est définie par la matrice jacobienne de l'image ou la matrice d'interaction $\mathbf{J}_s$. La synthèse de la commande repose sur l'élaboration d'une méthode de calcul explicite de cette matrice souvent associée à des primitives géométriques simples, telles que des points, des droites, des cercles, des ellipses, ou encore des moments de l'image.

L'objectif de tous les régimes de commande à base d'images est de minimiser une erreur $\mathbf{e}_s(t)$, qui est généralement définie par $\mathbf{e}_s(t) = \mathbf{s}_d(t) - \mathbf{s}(t)$. On a la loi de commande

$$\mathbf{v} = \widehat{\mathbf{J}_s^+}\mathbf{K}_s\mathbf{e}_s \tag{57}$$

où $\mathbf{J}_s^+$ est la pseudo-inverse de la matrice $\mathbf{J}_s$ et $\mathbf{K}$ est une matrice difinie-positive. Cette loi de commande assure que l'erreur $\mathbf{e}_s$ converge vers zéro de façon exponentielle avec un taux de convergence qui dépend du choix de $\mathbf{K}_s$. Le schéma bloc de commande est représenté sur la figure 5.

4.4.1 Calcul de la jacobienne de l'image

La matrice jacobienne qui correspond à un point $\mathbf{p}(X, Y)$ de l'image :

$$\mathbf{J}_s(\mathbf{s}, \varphi) = \begin{bmatrix} \frac{1}{H-H_o}(-Y\sin\varphi + XY\cos\varphi) & 1 + X^2 + \frac{T}{H-H_o}(Y\cos\varphi + XY\sin\varphi) \\ \frac{1}{H-H_o}Y^2\cos\varphi & XY + \frac{T}{H-H_o}Y^2\sin\varphi \end{bmatrix} \tag{58}$$

La matrice jacobienne de l'image d'un ensemble de k points de l'objet $\mathbf{p}_1, \cdots \mathbf{p}_k$ peut être construite en considérant $(2k \times 1)$ vecteurs de primitives. Si $\mathbf{J}_{\mathbf{s}_i}(\mathbf{s}_i, \varphi)$ désigne la

matrice jacobienne correspondant au point $\mathbf{p}_i$, alors la matrice jacobienne de l'ensemble des points sera la matrice d'ordre $(2k \times 2)$:

$$\mathbf{J}_s(\mathbf{s}, \varphi) = \begin{bmatrix} \mathbf{J}_{s1}(\mathbf{s}_1, \varphi) \\ \vdots \\ \mathbf{J}_{sk}(\mathbf{s}_k, \varphi) \end{bmatrix} \tag{59}$$

La matrice jacobienne d'un moment d'image d'ordre (i, j) est :

$$\mathbf{J}_{mij} = [J^v_{mij} \quad J^\omega_{mij}] \tag{60}$$

où

$$\begin{aligned} J^v_{mij} &= \frac{1}{H_m}((i+j+3)m_{i,j+1}\cos\varphi - im_{i,j}\sin\varphi) \\ J^\omega_{mij} &= im_{i-1,j} + (i+j+3)m_{i+1,j} + \\ &\quad \frac{T}{H_m}\left[im_{i-1,j+1}\cos\varphi + (i+j+3)\, m_{i,j+1}\sin\varphi\right] \end{aligned} \tag{61}$$

4.4.2 Planification de trajectoire dans le plan de l'image

Notre approche pour l'asservissement visuel 2D est basée sur la régulation d'une fonction d'erreur entre la mesure actuelle et une valeur constante désirée. Il n'est, par conséquent, pas évident de présenter des contraintes dans les trajectoires réalisées, telles que l'objet cible reste dans le champ de vue de la caméra, ou pour assurer la convergence de toutes les configurations initiales. En outre, nous avons utilisé l'approche de $\widehat{J_s^+} = J_s^+(s(t))$ qui nécessite le calcul de $J_s^+(s(t))$ à chaque itération, qui prend du temps et rend la matrice J_s, et donc le système, plus sensible au bruit d'image et aux erreurs de mesure. D'autre part, il semble utile d'utiliser l'approche $\widehat{J_s^+} = J_s^+(s_d)$ qui peut être calculée hors-ligne. Ceci, cependant, implique que la stabilité est assurée dans un petit voisinage de $\mathbf{s}_d$.

Le moyen de contourner ce problème consiste à effectuer une planification de trajectoire sur le plan de l'image, c'est-à-dire, fournir les valeurs désirées des primitives d'image $\mathbf{s}_d(t)$ qui correspondent à une trajectoire cartésienne réalisable. Avec ce couplage de la planification de trajectoire dans l'espace image et l'asservissement visuel à base d'images, les contraintes que l'objet reste dans le champ de la caméra peuvent être prises en compte au niveau de la planification. En outre, les mesures courantes des primitives d'image restent toujours proches de leurs valeurs désirées et la robustesse de l'asservissement visuel à base d'images est assurée le long de la trajectoire.

Un schéma bloc de cette approche est illustré sur la figure 6.

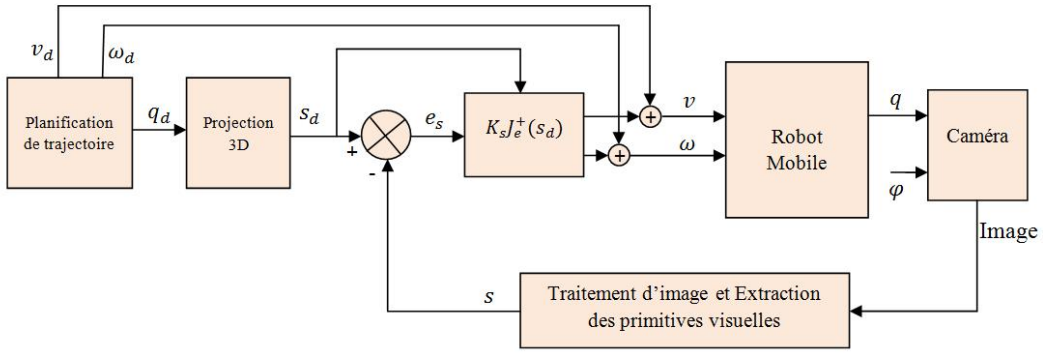


Figure 6: Schéma d'un Asservissement visuel 2D associée à une planification de trajectoire sur le plan image.

Chapitre 5 : Résultats expérimentaux

Les essais de validation expérimentale ont d'abord été réalisés sur le robot mobile Koala, qui est un robot à roues de taille moyenne. Il dispose de deux roues motorisées, avec une vitesse maximale de 0,4 m/s. Il est équipé de 16 capteurs de proximité et de capteurs de lumière ambiante, ainsi que d'une caméra montée sur une tourelle. Koala est muni d'un processeur Motorola 68331@22MHz. Une stratégie de commande basée sur les modes de glissement intégraux combinée avec une linéarisation par retour d'état a été mise en œuvre sur le Koala. Ce contrôleur a fourni un résultat bien meilleur que celui du régulateur PID linéaire traditionnel dans le cas de suivi de trajectoires et de rejet de perturbations.

La caméra fixe du Koala et le manque de qualité de l'image, le rendent peu commode pour mettre en œuvre les techniques d'asservissement visuel. Afin de valider nos résultats d'asservissement visuel, nous avons utilisé le robot mobile Pekee II. C'est aussi un robot à roues de taille moyenne, en forme de cylindre conçu pour des applications intérieures. Il a deux roues motorisées, avec une vitesse maximale de 250 mm/s et un couple maximal de 3.0Nm. Les positions des roues sont mesurées au moyen de deux odomètres (255 560 sommets par cycle de roue). Le robot est équipé de 8 capteurs ultrason et un pan-tilt AXIS 214 PTZ caméra qui offre une sortie vidéo avec une résolution jusqu'à 704x576 avec un débit d'images jusqu'à 30 images par seconde. Une stratégie de commande fondée sur l'application de la méthodologie de *l'Immersion et*

l'Invariance a été appliquée sur le robot pour la stabilisation et le suivi de trajectoires. Des résultats satisfaisants ont été obtenus pour les deux problèmes de commande. Une stratégie d'asservissement visuel basé sur la position a été mise en œuvre. Celle-ci a assuré un suivi de trajectoire satisfaisant, tant que l'objet cible reste dans le champ de la caméra, ce qui est assuré par le bon choix des paramètres de la trajectoire de référence. La différence entre la trajectoire calculée à l'aide de l'estimation de la pose et celle calculée en utilisant les odomètres est due aux incertitudes de calibrage de la caméra. Une stratégie d'asservissement visuel basé sur l'mage a été ensuite mise en œuvre en utilisant les moments d'image comme primitives visuelles, cette stratégie d'asservissement visuel est combinée avec une planification de trajectoires sur le plan de l'image, Les expériences ont montré un bon suivi de trajectoires.

Contributions

Les principales contributions de cette thèse portent sur les points qui suivent :

- Une stratégie de commande basée sur les modes de glissement intégraux est appliquée, avec l'objectif de la stabilisation et le suivi des trajectoires. Cette stratégie est combinée avec la linéarisation par retour d'état, qui est étendue pour inclure le modèle cinématique et le modèle dynamique, ou ce qu'on appelle le modèle cinématique de deuxième ordre. Cette combinaison a conduit à des résultats satisfaisants en termes de stabilité et de robustesse [29].

- Une stratégie de commande basée sur l'application de *l'immersion et l'Invariance* est développée, le système en boucle fermée se comporte asymptotiquement comme un système cible donné, ce qui rend le réglage des performances du système plus simple et avec une signification physique. La stabilité du système est garantie par un choix approprié des systèmes cibles.

- Deux méthodes permettant de conduire le robot mobile vers une position désirée à l'aide de l'asservissement visuel, l'un basé sur la position (3D), l'autre sur l'image (2D), ont été étudiées. Les moments d'image ont été utilisés comme primitives d'image. L'extraction des moments d'image s'est avérée plus robuste vis à vis du bruit d'image.

- Une nouvelle approche de l'asservissement visuel qui repose sur l'image est ici proposée. Elle est basée sur la génération de trajectoires sur le plan de l'image directement (Calcul des valeurs des primitives d'image correspondantes à une trajectoire cartésienne donnée). Cette approche garantit que la robustesse et la stabilité bien connues de l'asservissement 2D ont été étendues en raison du fait que les emplacements initial et désiré de la caméra sont proches. Les trajectoires obtenues garantissent que la cible reste dans le champ de vue de la caméra et que le mouvement du robot correspondant est physiquement réalisable [30].

Introduction

State of the art

The increased use of intelligent robotic systems in current indoor and outdoor applications is due to the efforts made by researchers and engineers on all fronts. Today, mobile systems have greater autonomy, and new applications abound, ranging from factory transport systems, airport transport systems, road/vehicular systems, to military applications, automated patrol systems, homeland security surveillance, and rescue operations.

Mobile robots may be classified by:

- The environment in which they move around:
  - Land robots. They are most commonly wheeled, but also include legged robots with two or more legs (humanoid, or resembling animals or insects) and robots on tracks.
  - Aerial robots are usually referred to as unmanned aerial vehicles (UAVs)
  - Underwater robots are usually called autonomous underwater vehicles (AUVs)
- The device they use to move, mainly:
  - Wheeled mobile robot (WMR).
  - Legged robot : human-like legs (i.e. an android) or animal-like legs.
  - Tracks.

We can estimate that the wheeled mobile robots are the bulk of mobile robots (one may see [20] for their classification and structural properties of their kinematic and dynamic models). Historically, their study came soon enough, following that of robot

manipulators in the mid 70s. Their low complexity has made good first subjects of study interested in robotics for autonomous systems. However, despite their apparent simplicity (flat mechanisms, linear actuators), these systems have raised a number of difficult problems. A number of these problems arise from the fact that these systems are characterized by kinematic constraints that are not integrable and cannot therefore be eliminated from the model equations, these constraints are the so-called nonholonomic constraints. These constraints typically arise when the system has less controls than configuration variables. For instance a differential-drive robot has two controls (linear and angular velocities) while it moves in a 3-dimensional configuration space, see for instance [73]. As a consequence, any path in the configuration space does not necessarily correspond to a feasible path for the system. This is basically why the purely geometric techniques developed in motion planning for holonomic systems do not apply directly to nonholonomic ones. In fact, as pointed out in an early paper of Brockett [17], nonholonomic control systems with more degrees of freedom than controls (underactuated mechanisms), though controllable in their configuration space ([15], [10], [99]), cannot be stabilized by any C^1 state feedback control.

The above-mentioned peculiar nature of nonholonomic kinematics in addition to the growing applications of wheeled mobile robots has attracted interest of researchers. Many studies in nonholonomic control systems have been carried out in past decades, e.g., see([58], [86], [16], [103], [4], [25],[33], [34], [96]). In the case of wheeled mobile robots it is possible to distinguish between two control problems: Stabilization about a certain robot posture and tracking a given trajectory. From a control viewpoint, the special nature of nonholonomic kinematics makes the second problem easier than the first; in fact, this problem is truly nonlinear; linear control is ineffective, even locally, and innovative design techniques are needed.

Many control strategies have been proposed in the literature to solve both control problems, most of the control laws are based on the kinematic model of the mobile robot and are not applicable for systems where forces and torques are considered as the inputs since these controllers are not differentiable. Discontinuous state feedback controller is used in [4], [5]. Time-varying controllers have been designed to stabilize mobile robot in ([85], [32], [54], [55]). Among those, a global exponential set point control is proposed in [32] and bounded tracking control in [55]. Backstepping based methods have been considered in several papers ([39]; [54]; [98]) a switched finite-time control

algorithm has been proposed in [9] and dynamic feedback linearization has been used for trajectory tracking and posture stabilization in [77]. Stabilization of the dynamic WMRs has received lesser attention in the literature. A discontinuous controller has been developed in [6] for the kinematic and dynamic model of the mobile robot using a polar coordinates system. In [38] a combined kinematic/torque control law is developed using backstepping and asymptotic stability is guaranteed by Lyapunov theory.

The notion of planning a trajectory for robot manipulators has appeared at the early sixties, but a real progress in that field had to wait till early eighties when many works in the fields of robotics and mathematics algorithms were used to develop and implement a number of algorithms for trajectory planning. One may refer to [61] and [88] for more details on path and trajectory planning. The easier and more interesting (from an engineering perspective) task of tracking a given trajectory has received much attention; in [58], [76], [68], [38] a local viewpoint in the stabilization feedback design has been taken by using Taylor linearization of the corresponding error model. A dynamic feedback approach has been proposed in [28]. Tracking control using Lyapunov's direct method can be seen in [25], [54].

The adoption of sensors is of crucial importance to achieve high-performance robotic systems. It is worth classifying sensors into *proprioceptive* sensors that measure the internal state of the manipulator, and *exteroceptive* sensors that provide the robot with knowledge of the surrounding environment. Exteroceptive sensors include force sensors, proximity sensors, range sensors and vision sensors. Vision sensors, or more precisely video cameras, are exteroceptic sensors that provide the richest information. So it is natural that many studies have examined the coupling of vision and robotics. This area of research became known as visual servoing since it comes to controlling the movement of a robot using visual information. Many works are based on the exploitation of visual data to achieve various objectives such as positioning against an object, monitoring, seizure, etc.. The first use of vision in closed loop is due to Shirai and Inoue [91] who described how a vision sensor can increase positioning accuracy. There was talk of visual feedback. But it is to Park and Hill [47] that we owe the appearance of the term visual servoing. Visual servoing techniques could be classified into three main classes [51]: Position Based Visual Servoing (PBVS or 3D), Image Based Visual Servoing (IBVS or 2D) and the Homography Based Visual Servoing (HBVS or 2 1/2 D) (see [31] for the application of visual servoing techniques on an UAV).

Position based visual servoing needs a full reconstruction of the target pose with respect to the camera; it leads to a state estimation problem in the Cartesian frame [95], [40], and a classical state-space control design [3], [90], [87]. The main drawback of the PBVS methods is the need of a perfect knowledge of the target geometrical model as described in [51], hence it is highly sensitive to camera calibration errors. The second class, known as 2D visual servoing, aims to control the dynamics of features directly in the image plane. Many extensions to the classical IBVS methods have been proposed for the control of non-linear dynamic systems, as the robust backstepping based approach proposed in [44], [108], and optimal control techniques [109]. Image moments have been widely used in computer vision for a very long time, especially for pattern recognition applications, see e.g. [50], [82], [70], or in 2D visual servoing of robotic systems in many articles [12], [22], [23]. A path planning in image plane for an eye-in-hand robotic system was proposed in [67], and for a mobile robot in [84]. Both papers used points as visual features.

Context and Objectives

Autonomous robots are robots that can perform desired tasks in unstructured environments without continuous human guidance. The development of an autonomous robot poses many fundamental problems in various fields. For example, the problem of perception of the environment by the robot (sensors) is related to the field of vision, the problem of trajectory tracking is related to control theory, and another fundamental problem is that of trajectory planning. In other words, an autonomous robot must be able to sense the environment boundary and obstacles, decide how to move from some point to the other (motion planning), and finally, control its driving mechanism such that the planned motion is actually executed in reality.

The objective of this thesis is to design and implement the methods that give a mobile robot the ability to perform a task or achieve a given mission. For this, it is necessary to determine robot's movements relative to its local environment. These movements are performed using the trajectory planning, obstacle avoidance, motion control, localization, perception and navigation. More precisely, within the scope of this thesis, we will be interested in the following methodological aspects:

- Reactive path planning in order to find a collision-free path from a given initial position to a predefined target point in presence of static obstacles.
- Deriving discontinuous state feedback controllers to address both nonholonomic robot motion control problems: Stabilization about a certain robot posture and tracking a given trajectory.
- Proposing strategies to enable a mobile robot to perform a given task independently, that is to say solely from information provided in real time by sensors. A very important step of this advanced autonomy is visual servoing. So we will be interested in the problem of the control and achievement of visual-based tasks for the navigation of a mobile robot and the execution and control of basic motions like going to an object or target following.
- Conducting simulations and experimental tests on a wheeled mobile robot to establish validity for the proposed strategies.

Thesis Outline

This document is composed of five chapters. In the first chapter, we present the modeling of the mobile robot of type differential-drive, we begin by describing robot position in the configuration space. The structure of the kinematic constraints arising from the pure rolling of the wheels is then analyzed; it is shown that such constraints are in general nonholonomic and consequently reduce the local mobility of the robot. The kinematic model associated with the constraints is introduced to describe the instantaneous admissible motions, control properties of this model have been shown, and conditions are given under which it can be put in chained form. Finally, the dynamic model of the mobile robot is presented at the end of this chapter.

The second chapter is dedicated to the trajectory planning problem in the presence of nonholonomic constraints. The existence of flat outputs is exploited to devise trajectory planning methods that guarantee that the nonholonomic constraints are satisfied. Path planning in presence of obstacle is then discussed, with the goal of generating smooth collision-free trajectories.

In chapter three, we will discuss the motion control problem for a nonholonomic mobile robot, with reference to two basic motion tasks, i.e., posture regulation and

trajectory tracking. First, we will discuss a robust control strategy based on Integral Sliding Mode controller combined with state feedback linearization. In addition, we will study the method known as *Inversion and Immersion* developed by A. Astolfi and R. Ortega [8] to assess the degree of applicability and performance on nonholonomic robot systems in relation to more usual methods in nonlinear control theory. Simulations will be performed to assess the quality of both control techniques.

In chapter four, we address visual servoing for a mobile robot, we begin by presenting the modelling of the (camera + robot) system, then we present some basic algorithms for image processing, aimed at extracting numerical information referred to as image feature parameters (mainly image moments). These parameters, relative to images of objects present in the scene observed by a camera, can be used to estimate the pose of the camera with respect to the objects and vice versa. To this end, analytic pose estimation methods, based on the measurement of a certain number of points are presented. A fundamental operation concerns the camera calibration; to this end, a calibration method based on the measurement of a certain number of correspondences is presented. Then, the two main approaches to visual servoing are introduced, namely position-based visual servoing and image-based visual servoing. Using the so-called image moments as image feature parameters in both approaches is also presented. The novel idea here, as far as we know, is to perform 2D visual servoing using image moments as visual features, while planning the trajectory in the image plane, instead of the Cartesian space as usual in the literature. Chapter five is then devoted to experimental results of motion control and visual servoing techniques on two mobile robots: Koala and Pekee II in order to assess the quality and applicability of those techniques. Experimental tests have shown the validity of our approach.

We conclude the document with some final remarks and some perspectives.

Contributions

The main contributions of this thesis are being made on the following points:

- A robust control strategy on the basis of Integral Sliding Mode approach is derived, with the objective of posture regulation and tracking pre-generated trajectories. This controller is combined with state feedback linearization of both

kinematic and dynamic models. This combination has led to satisfactory results in terms of stabilization and robustness [29].

- A control strategy based on the application of *Immersion and Invariance* methodology is described. The proposed control strategy guarantees that the closed-loop system asymptotically behaves like a given target system achieving asymptotic model matching, which makes the system performance adjustment simpler and physically meaningful. Stability of the system has been guaranteed by appropriate choice of the target systems.

- Presenting two methods devoted to drive a mobile robot to a desired position using visual servoing. Both Position-based visual servoing and Image-based visual servoing have been examined. In Image-based visual servoing, image moments have been used as image features, the extraction of image moments from the image proved to be more robust with respect to image noise and varying luminosity .

- A new approach to image-based visual servoing is developed, which is based on the trajectory generation on the image plane directly (computing image features values corresponding to a given Cartesian trajectory). This approach guarantees that the well-known robustness and stability of image-based servoing have been extended due to the fact that initial and desired camera locations are close to each other. The obtained trajectories ensure that the target remains in the camera field of view and that the corresponding robot motion is physically realizable [30].

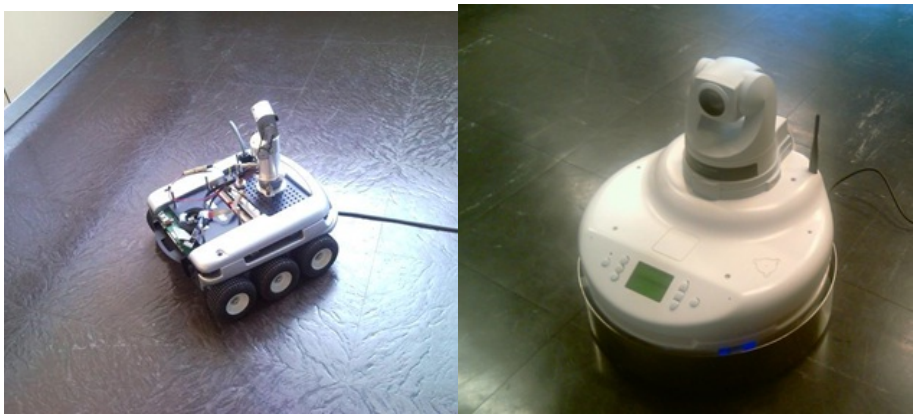
Chapter 1

Modelling

In this chapter we present the modelling of wheeled mobile robots. The kinematic model of a unicycle is very known, but for the reader's convenience, we will remind of the main steps. The structure of the kinematic constraints arising from the pure rolling of the wheels is first analyzed; it is shown that such constraints are in general nonholonomic and consequently reduce the local mobility of the robot. The kinematic model associated with the constraints is introduced to describe the instantaneous admissible motions, and conditions are given under which it can be put in chained form. Finally, the dynamic model, that relates the admissible motions to the generalized forces acting on the robot DOFs, is presented at the end of this chapter.

1.1 Kinematic Models and Constraints

Deriving a model for the whole robot's motion is a bottom-up process. Each individual wheel contributes to the robot's motion and, at the same time, imposes constraints on it. Wheels are tied together based on robot chassis geometry, and therefore their constraints combine to form constraints on the overall motion of the robot chassis. But the forces and constraints of each wheel must be expressed with respect to a clear and consistent reference frame. In the following, the kinematic model of a wheeled vehicle of type differential-drive will be analyzed in detail.



(a) KOALA **(b)** PEKEE II

Figure 1.1: Examples of Differential-drive mobile robots

1.1.1 Differential-drive (Hilare) Mobile Robot

This type of robot is very popular because of its simplicity of construction and interesting kinematic properties. Figure 1.1 shows two differential-drive mobile robots. A schematic figure of a Differential-drive mobile robot is shown in Fig. 1.2. This type of robot is mostly used for indoor applications. Its drive mechanism has two independent motors; each of these motors powers one of the robot's wheels. Thus, the actual kinematic inputs that drive the robot and affect its speed and direction of motion are the two wheel speeds. With this in mind, at first glance it seems intuitive to write the kinematic equations of motion of a differential-drive mobile robot in terms of these speeds. However, on most commercial mobile robots, there exists a low-level controller that controls the linear and angular velocity of the robot. Therefore, for application purposes, it is more convenient to choose the linear and angular velocity of the mobile robot as the inputs of the kinematic model.

Let's consider the differential-drive mobile robot shown in Fig. 1.2. We assume that the robot motion is reasonably slow such that the longitudinal traction and lateral force exerted on the robot's tires do not exceed the maximum static friction between the tires and the floor in the longitudinal and lateral directions. In other words, we assume that no slip happens between the robot's tires and the floor during the whole motion of the robot.

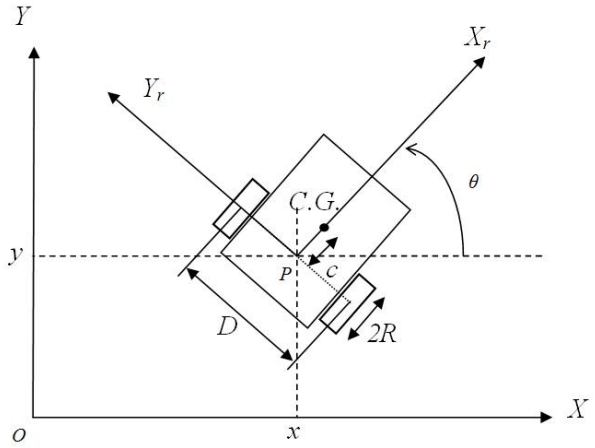


Figure 1.2: Differential-drive mobile robot.

The first direct result of this assumption is that the velocities of the center of the robot's wheels do not have any lateral components. As a consequence, one can assume that the velocity of point (x, y), the midpoint of the line attaching the center of the wheels, does not have any lateral component and is parallel to the wheel planes. The second result of the no-slip assumption is that one can relate the velocity of point (x, y) to the rotational velocity of the wheels.

Let the coordinates of point $p(x, y)$ define the global position of the robot with respect to the inertial coordinate system $O : \{X, Y\}$. We consider a line that is perpendicular to the wheel axis and goes through the point (x, y) as an orientation reference for the robot. The angle that this line makes with the positive X axis, θ, represents the orientation of the robot.

We assume that the point (x, y) on the robot moves with a linear speed of v, while the robot has an angular velocity of ω. Now, we can use the first direct result of the no-slip assumption and write the velocity components of the point (x, y) in the inertial frame as

$$\begin{aligned} \dot{x} &= v \cos\theta \\ \dot{y} &= v \sin\theta \end{aligned} \tag{1.1}$$

Also, the rate of change of the robot's orientation is

$$\dot{\theta} = \omega \tag{1.2}$$

Combining Eqs. (1.1) and (1.2) results in the kinematic equations of motion of the robot, which can be written in the following matrix form

$$\dot{\mathbf{q}} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \mathbf{v} \tag{1.3}$$

where $\mathbf{v} = [v \quad \omega]^T$. We note that the differential-drive mobile robot is kinematically equivalent to a unicycle, which is a vehicle with a single orientable wheel[1]. We can also put eq. (1.3) in the form

$$\dot{\mathbf{q}} = \mathbf{G}(\mathbf{q})\mathbf{v} = \mathbf{g}_1(\mathbf{q})v + \mathbf{g}_2(\mathbf{q})\omega \tag{1.4}$$

where $\mathbf{g}_1$ and $\mathbf{g}_2$ are the two input vector fields.

The driving and steering velocities v and ω can be expressed as a function of the actual velocity inputs, i.e., the angular speeds ω_r and ω_l of the right and left wheel, respectively:

$$\begin{aligned} v &= \frac{R(\omega_r + \omega_l)}{2} \\ \omega &= \frac{R(\omega_r - \omega_l)}{D} \end{aligned} \tag{1.5}$$

where R is the radius of the wheels and D is the distance between their centers.

The constraint that the wheel cannot slip in the lateral direction is given in form (A.3) as (see appendix A)

$$\dot{x}\sin\theta - \dot{y}\cos\theta = [\sin\theta \quad -\cos\theta \quad 0]\,\dot{\mathbf{q}} = 0 \tag{1.6}$$

In this case, holonomy condition (A.8) gives:

$$\frac{\partial\gamma}{\partial y}\sin\theta = -\frac{\partial\gamma}{\partial x}\cos\theta \tag{1.7}$$

$$\gamma\cos\theta + \frac{\partial\gamma}{\partial\theta}\sin\theta = 0 \tag{1.8}$$

[1]From now on we will refer to the differential-drive (Hilare-type) mobile robot by unicycle

$$\gamma \sin\theta - \frac{\partial \gamma}{\partial \theta}\cos\theta = 0 \tag{1.9}$$

By proper multiplication and addition of (1.8) and (1.9) we get $\partial\gamma/\partial\theta = 0$ and $\gamma = 0$. This confirms that constraint (1.6) is nonholonomic.

System (1.3) displays a number of structural control properties (see [18], [27], [20])

A. Controllability at a point

The local linearization of eq. (1.3) at any point $\mathbf{q}_e$ is

$$\dot{\tilde{\mathbf{q}}} = \begin{bmatrix} \cos\theta_e & 0 \\ \sin\theta_e & 0 \\ 0 & 1 \end{bmatrix} \mathbf{v} = \mathbf{G}(\mathbf{q}_e)\mathbf{v} \qquad\qquad \tilde{\mathbf{q}} = \mathbf{q} - \mathbf{q}_e \tag{1.10}$$

The rank of the controllability matrix of system (1.10) is two, which means that the local linearized system is not controllable. In order to study the controllability of the unicycle, we use tools from nonlinear control theory [52]. Since the system is driftless (i.e., no motion takes place under zero input), a more systematic approach is to take advantage of the controllability conditions for nonlinear driftless systems. In particular, controllability may be verified using the accessibility rank condition.

$$\dim \Delta_{\mathcal{A}} = n \tag{1.11}$$

where $\Delta_{\mathcal{A}}$ is the accessibility distribution associated with system (1.4), i.e., the involutive closure of distribution $\Delta = span\{\mathbf{g}_1, \mathbf{g}_2\}$

It is easy to check that the accessibility rank condition is satisfied globally (at any q_e), since

$$\dim \Delta_{\mathcal{A}} = \operatorname{rank}([\mathbf{g}_1 \quad \mathbf{g}_2 \quad [\mathbf{g}_1, \mathbf{g}_2]]) = \operatorname{rank}(\begin{bmatrix} \cos\theta & 0 & \sin\theta \\ \sin\theta & 0 & -\cos\theta \\ 0 & 1 & 0 \end{bmatrix}) = 3 = n \tag{1.12}$$

with $[\mathbf{g}_1, \mathbf{g}_2]$ being Lie bracket of the two input vector fields, and since the system is driftless , condition (1.12) implies its controllability.

Controllability can also be shown constructively, i.e., by providing an explicit sequence of maneuvers bringing the robot from any start configuration (x_0, y_0, θ_0) to any desired goal configuration (x_d, y_d, θ_d). Since the unicycle can rotate on itself, this task

is simply achieved by an initial rotation on (x_0, y_0) until the unicycle is oriented toward (x_d, y_d), followed by a translation to the goal position, and by a final rotation on (x_d, y_d) so as to align θ with θ_d.

However, as for the stabilizability to a point, system (1.3) has less control inputs than generalized coordinates, so it fails to meet Brockett's necessary condition for smooth stabilizability of a driftless regular systems (i.e., such that the input vector fields are well defined and linearly independent at q_e), which requires a number of inputs equal to the number of states (see Appendix A.2).

B. Controllability about a trajectory

In order for the robot to perform a given desired cartesian motion, it is more convenient to generate a corresponding state trajectory $\mathbf{q}_d(t) = [x_d(t), y_d(t), \theta_d(t)]^T$. In order for this trajectory to be feasible, it must satisfy the nonholonomic constraint on the vehicle motion or, in other words, it must satisfy the equations

$$\begin{array}{rcl} \dot{x}_d & = & v_d \cos\theta_d \\ \dot{y}_d & = & v_d \sin\theta_d \\ \dot{\theta}_d & = & \omega_d \end{array} \tag{1.13}$$

for some choice of reference inputs v_d and ω_d.

It is possible to compute the error vector by comparing the desired state $\mathbf{q}_d(t) = [x_d(t), y_d(t), \theta_d(t)]^T$ with the current measured state $\mathbf{q}(t) = [x(t), y(t), \theta(t)]^T$. However, a simpler analysis can be conducted by defining the state tracking error through a rotation matrix as [57], [74].

$$\mathbf{e} = \begin{bmatrix} \mathbf{e}_1 \\ \mathbf{e}_2 \\ \mathbf{e}_3 \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_d - x \\ y_d - y \\ \theta_d - \theta \end{bmatrix} \tag{1.14}$$

By differentiating $\mathbf{e}$ with respect to time, and using (1.3) and (1.13), one easily finds

$$\begin{aligned} \dot{e_1} &= v_d \cos e_3 - v + e_2\omega \\ \dot{e_2} &= v_d \sin e_3 - e_1\omega \\ \dot{e_3} &= \omega_d - \omega \end{aligned} \tag{1.15}$$

Using the input transformation

$$v = v_d \cos e_3 - u_1 \tag{1.16}$$

$$\omega = \omega_d - u_2 \tag{1.17}$$

which is clearly invertible, the following expression is obtained for the tracking error dynamics

$$\dot{\mathbf{e}} = \begin{bmatrix} 0 & \omega_d & 0 \\ -\omega_d & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \mathbf{e} + \begin{bmatrix} 0 \\ \sin e_3 \\ 0 \end{bmatrix} \cdot v_d + \begin{bmatrix} 1 & -e_2 \\ 0 & e_1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{1.18}$$

Let's consider local linearization of system (1.18) about the reference trajectory, on which clearly $\mathbf{e} = 0$.

$$\dot{\mathbf{e}} = \begin{bmatrix} 0 & \omega_d & 0 \\ -\omega_d & 0 & v_d \\ 0 & 0 & 0 \end{bmatrix} \mathbf{e} + \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{1.19}$$

When v_d and ω_d are constant, the above linear system becomes time-invariant and controllable, since matrix

$$\mathcal{C} = \begin{bmatrix} B & AB & A^2B \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & -\omega_d^2 & v_d\omega_d \\ 0 & 0 & -\omega_d & v_d & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{1.20}$$

has rank 3 provided that either v_d or ω_d are nonzero. Therefore, we conclude that kinematic system (1.3) can be locally stabilized by linear feedback about trajectories which consist of linear or circular paths, executed with constant velocity [27].

C. Flat Outputs

The flatness property is very essential to solve the trajectory planning problem more efficiently. A nonlinear dynamic system $\dot{\mathbf{x}} = f(\mathbf{x}) + G(\mathbf{x})\mathbf{u}$ is *differentially flat* if there exists a set of outputs $\mathbf{y}$, called flat outputs, such that the state $\mathbf{x}$ and the control inputs $\mathbf{u}$ can be expressed algebraically as a function of $\mathbf{y}$ and its time derivatives up to a certain order

$$\mathbf{x} = \mathbf{x}(\mathbf{y}, \dot{\mathbf{y}}, \ddot{\mathbf{y}}, \cdots, \mathbf{y}^{(r)})$$

$$\mathbf{u} = \mathbf{u}(\mathbf{y}, \dot{\mathbf{y}}, \ddot{\mathbf{y}}, \cdots, \mathbf{y}^{(r)})$$

As a consequence, once an output trajectory is assigned for $\mathbf{y}$, the associated trajectory of the state $\mathbf{x}$ and history of control inputs $\mathbf{u}$ are uniquely determined.

In the case of unicycle, let's refer to kinematic model (1.3). Its first two equations imply that, given a Cartesian path $(x(t), y(t))$, the associated state trajectory is $\mathbf{q}(t) = [x(t)\ y(t)\ \theta(t)]^T$ where

$$\theta(t) = \text{atan2}(\dot{y}(t), \dot{x}(t)) + k\pi \qquad k = 0, 1. \tag{1.21}$$

where atan2 is a two aruments function (See A.3 for the definition of atan2).

The two possible choices for k account for the fact that the same Cartesian path may be followed moving forward ($k = 0$) or backward ($k = 1$). If the initial orientation of the robot is assigned, only one of the choices for k is correct. The geometric inputs that drive the robot along the Cartesian path are easily obtained from (1.3),(1.21) as

$$v(t) = \pm\sqrt{(\dot{x}(t))^2 + (\dot{y}(t))^2} \tag{1.22}$$

$$\omega(t) = \frac{\dot{x}(t)\ddot{y}(t) - \ddot{x}(t)\dot{y}(t)}{(\dot{x}(t))^2 + (\dot{y}(t))^2} \tag{1.23}$$

where the choice of the sign of $v(t)$ depends on the type of motion (forward or backward).

1.2 Chained Form

The possibility of transforming kinematic model (1.3) of a mobile robot in a canonical form is of great interest for solving planning and control problems with efficient, systematic procedures. The most useful canonical structure is the *chained form*.

A $(2, n)$ chained form is a two-input driftless system

$$\dot{z} = \gamma_1(z)u_1 + \gamma_2(z)u_2$$

whose equations are expressed as

$$\begin{aligned} \dot{z}_1 &= u_1 \\ \dot{z}_2 &= u_2 \\ \dot{z}_3 &= z_2 u_1 \\ &\vdots \\ \dot{z}_n &= z_{n-1} u_1 \end{aligned} \tag{1.24}$$

Using the following notation for a 'repeated' Lie bracket

$$\begin{aligned} ad_f^k g(x) &= \left[f, ad_f^{k-1} g\right](x) \\ ad_f^0 g(x) &= g(x) \\ ad_f g(x) &= [f, g](x) \end{aligned}$$

one has for system (1.24)

$$\gamma_1 = \begin{bmatrix} 1 \\ 0 \\ z_2 \\ z_3 \\ \vdots \\ z_{n-1} \end{bmatrix}, \quad \gamma_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad \Rightarrow ad_{\gamma_1}^k \gamma_2 = \begin{bmatrix} 0 \\ \vdots \\ (-1)^k \\ \vdots \\ 0 \end{bmatrix} \tag{1.25}$$

where $(-1)^k$ is the $(k+2)th$ component. This implies that the system is controllable, because the accessibility distribution

$$\Delta_{\mathcal{A}} = span\left\{\gamma_1, \gamma_2, ad_{\gamma_1}\gamma_2, \cdots, ad_{\gamma_1}^{n-1}\gamma_2\right\}$$

has dimension n. And we have that the degree of nonholonomy is $\kappa = n - 1$.

As mentioned in [71] and[59], many nonlinear mechanical systems can be transformed via coordinates change and feedback into chained form. We are interested in transforming a generic two-input driftless system

$$\dot{\mathbf{q}} = \mathbf{g}_1(\mathbf{q})u_1 + \mathbf{g}_2(\mathbf{q})u_2 \tag{1.26}$$

in chained form (1.24). Necessary and sufficient conditions have been given in [71] for the conversion of a two-input system like (1.26) into chained form by means of a change of coordinates and an invertible input transformation:

$$\mathbf{z} = T(\mathbf{q}) \qquad \mathbf{u} = B(\mathbf{q})\mathbf{v} \tag{1.27}$$

Let's consider kinematic model (1.4) of the unicycle. With the change of coordinates

$$\begin{aligned} z_1 &= \theta \\ z_2 &= x\cos\theta + y\sin\theta \\ z_3 &= x\sin\theta - y\cos\theta \end{aligned} \tag{1.28}$$

and the input transformation

$$\begin{aligned} v &= u_2 + z_3 u_1 \\ \omega &= u_1 \end{aligned} \tag{1.29}$$

which is clearly invertible, we obtain the (2,3) chained form (see [72] for details)

$$\begin{aligned} \dot{z}_1 &= u_1 \\ \dot{z}_2 &= u_2 \\ \dot{z}_3 &= z_2 u_1 \end{aligned} \tag{1.30}$$

We note that, while z_1 is the orientation θ, coordinates z_2 and z_3 represent the position of the robot in the moving body coordinate system $P\{X_r, Y_r\}$. This transformation is defined everywhere in the configuration space, with the exception of points where $\cos\theta = 0$. The equivalence between the two models is then subject to the condition $\theta \neq \pm k\pi/2$, with $k = 1, 2, \ldots$

It is easy to prove that the flat outputs of a $(2, n)$ chained form are z_1 and z_n, from which it is possible to compute all the other state variables as well as the associated control inputs. For example, in the case of the (2, 3) chained form (1.30) we can write

$$\begin{aligned} z_2 &= \frac{\dot{z}_3}{\dot{z}_1} \\ u_1 &= \dot{z}_1 \\ u_2 &= \frac{\dot{z}_1 \ddot{z}_3 - \ddot{z}_1 \dot{z}_3}{\dot{z}_1^2} \end{aligned}$$

In the next section we will discuss the dynamic model of the unicycle which is less frequent in literature.

1.3 Dynamic Model

The aim of this section is the derivation of a dynamical state-space model of wheeled mobile robot describing the dynamical relations between the configuration coordinates and the torques generated by the motors. We will consider that the robot is moving with small velocities which will allow us to neglect friction and Coriolis effects.

1.3.1 Equations of Motion

To find the equations of motion, we will use the Newton-Euler method in finding the accelerations of the centers of mass for all bodies. Here, the robot as a dynamic system

consists of three rigid bodies: the robot body and two wheels. For this purpose we will use the body coordinate system to express the inertial accelerations of the three bodies [36].

Let us assume that the inertial acceleration of the origin of the body coordinate system is

$$\mathbf{a}_0 = \begin{bmatrix} \ddot{x}_r \\ \ddot{y}_r \\ 0 \end{bmatrix}_R \tag{1.31}$$

and angular velocity and acceleration of the body coordinate system are

$$\boldsymbol{\omega} = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta} \end{bmatrix}_R, \qquad \boldsymbol{\alpha} = \begin{bmatrix} 0 \\ 0 \\ \ddot{\theta} \end{bmatrix}_R \tag{1.32}$$

Let's denote by $\mathbf{c} = [c\ 0\ 0]_R^T$ the distance between the center of mass of the robot's body and the origin of the robot's body frame. The inertial acceleration of the center of mass of the robot body expressed in the body coordinate system is

$$\begin{aligned} \mathbf{a}_r &= \mathbf{a}_0 + \boldsymbol{\alpha} \times \mathbf{c} - \omega^2 \mathbf{c} \\ \begin{bmatrix} a_{rx} \\ a_{ry} \\ 0 \end{bmatrix}_R &= \begin{bmatrix} \ddot{x}_r - c\dot{\theta}^2 \\ \ddot{y}_r + c\ddot{\theta} \\ 0 \end{bmatrix}_R \end{aligned} \tag{1.33}$$

Let's denote by $\mathbf{d}_r = [0\ \ -D/2\ 0]_R^T$ the distance between the center of mass of the robot's right wheel and the origin of the robot's body frame, then the inertial acceleration of the center of mass of the robot right wheel expressed in the body frame is

$$\begin{aligned} \mathbf{a}_{wr} &= \mathbf{a}_0 + \boldsymbol{\alpha} \times \mathbf{d}_r - \omega^2 \mathbf{d}_r \\ \begin{bmatrix} a_{wrx} \\ a_{wry} \\ 0 \end{bmatrix}_R &= \begin{bmatrix} \ddot{x}_r + \frac{D}{2}\ddot{\theta} \\ \ddot{y}_r + \frac{D}{2}\dot{\theta}^2 \\ 0 \end{bmatrix}_R \end{aligned} \tag{1.34}$$

and the same for the left wheel

$$\begin{aligned} \mathbf{a}_{wl} &= \mathbf{a}_0 + \boldsymbol{\alpha} \times \mathbf{d}_l - \omega^2 \mathbf{d}_l \\ \begin{bmatrix} a_{wlx} \\ a_{wly} \\ 0 \end{bmatrix}_R &= \begin{bmatrix} \ddot{x}_r - \frac{D}{2}\ddot{\theta} \\ \ddot{y}_r - \frac{D}{2}\dot{\theta}^2 \\ 0 \end{bmatrix}_R \end{aligned} \tag{1.35}$$

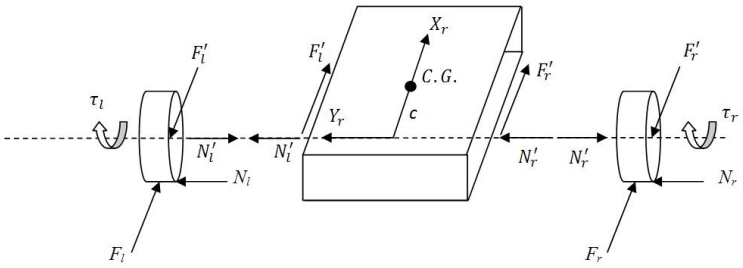
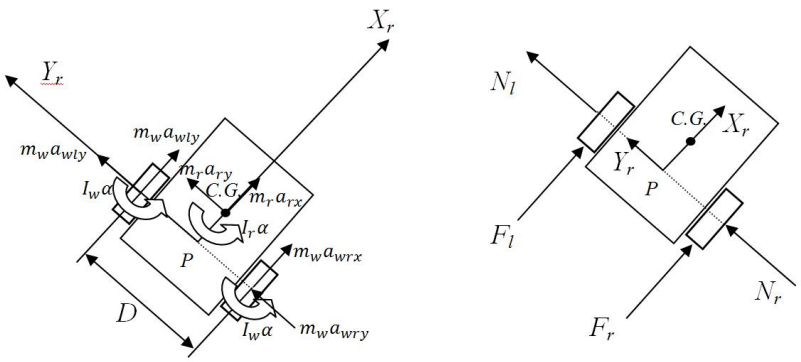


Figure 1.3: Free body diagram

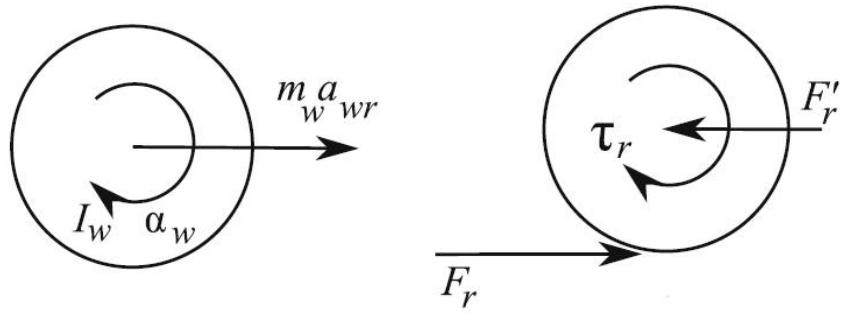


(a) Accelerations **(b)** Forces

Figure 1.4: Accelerations and forces diagram

From Fig. 1.3 and Fig. 1.4, one can write the external force and the inertia force balance in the X_r direction

$$\begin{aligned} F_r + F_l &= m_w a_{wrx} + m_w a_{wlx} + m_r a_{rx} \\ &= m_w \left(\ddot{x}_r + \frac{D}{2}\ddot{\theta} \right) + m_w \left(\ddot{x}_r - \frac{D}{2}\ddot{\theta} \right) + m_r \left(\ddot{x}_r - c\dot{\theta}^2 \right) \\ &= (m_r + 2m_w)\ddot{x}_r - m_r c\dot{\theta}^2 \end{aligned} \tag{1.36}$$



(a) Accelerations **(b)** Forces

Figure 1.5: Accelerations and forces diagram for the right wheel

The inertia force balance in the Y_r direction is given by

$$\begin{aligned} N_r + N_l &= m_w a_{wry} + m_w a_{wly} + m_r a_{ry} \\ &= m_w \left(\ddot{y}_r + \frac{D}{2}\dot{\theta}^2 \right) + m_w \left(\ddot{y}_r - \frac{D}{2}\dot{\theta}^2 \right) + m_r \left(\ddot{y}_r + c\ddot{\theta} \right) \\ &= (m_r + 2m_w)\ddot{y}_r + m_r c\ddot{\theta} \end{aligned} \tag{1.37}$$

The external moment and the inertia moment balance in the Z_r direction is

$$\begin{aligned} (F_r - F_l)\frac{D}{2} &= (I_r + 2I_w)\,\ddot{\theta} + m_w a_{wrx}\frac{D}{2} - m_w a_{wlx}\frac{D}{2} + m_r a_{ry} c \\ &= (I_r + 2I_w)\,\ddot{\theta} + m_w \left(\ddot{x}_r + \frac{D}{2}\ddot{\theta} \right)\frac{D}{2} - m_w \left(\ddot{x}_r - \frac{D}{2}\ddot{\theta} \right)\frac{D}{2} \\ &\quad + m_r c(\ddot{y}_r + c\ddot{\theta}) \\ &= \left(I_r + 2I_w + m_w \frac{D^2}{2} \right)\ddot{\theta} + m_r c(\ddot{y}_r + c\ddot{\theta}) \end{aligned} \tag{1.38}$$

By observing Fig. 1.5a and Fig. 1.5b, one can write the external moment and the inertia moment balance in the Y_r direction

$$\begin{aligned} -F_r R + \tau_r &= I_{wy}\alpha_w \\ &= I_{wy}\frac{1}{R}\left(\ddot{x}_r + \frac{D}{2}\ddot{\theta} \right) \end{aligned} \tag{1.39}$$

This results in the following for the traction force, F_r , on the right wheel

$$F_r = \frac{1}{R}\left(\tau_r - \frac{I_{wy}}{R}\left(\ddot{x}_r + \frac{D}{2}\ddot{\theta}\right)\right) \tag{1.40}$$

A similar result for the traction force of the left wheel, F_l , can be achieved

$$F_l = \frac{1}{R}\left(\tau_l - \frac{I_{wy}}{R}\left(\ddot{x}_r - \frac{D}{2}\ddot{\theta}\right)\right) \tag{1.41}$$

where τ_r and τ_l are the right and the left wheel's driving torque, respectively. Since these torques are the input to the system, we shall write the equations of motion in terms of these torques. This goal is achieved by substituting Eqs. (1.40) and (1.41) into Eqs. (1.36), (1.37), and (1.38). The resulting dynamic equations of motion are

$$\begin{aligned}
&\left(m_r + 2m_w + \frac{2I_{wy}}{R^2}\right)\ddot{x}_r - m_r c\dot{\theta}^2 = \frac{1}{R}(\tau_r + \tau_l) \\
&\left(m_r + 2m_w\right)\ddot{y}_r + m_r c\ddot{\theta} = N_r + N_l \\
&\left(I_r + 2I_w + m_w\frac{D^2}{2} + I_{wy}\frac{D^2}{2R^2} + m_r c^2\right)\ddot{\theta} + m_r c\ddot{y}_r = \frac{D}{2R}(\tau_r - \tau_l)
\end{aligned} \tag{1.42}$$

The lateral tire force $N_r + N_l$ can be determined by finding a kinematic relation that allow as to compute the lateral acceleration $\ddot{y}_r$. The nonholonomic constraint defined in Eq. (1.6) can be used for determining the lateral acceleration component $\ddot{y}_r$. The nonholonomic constraint is repeated here.

$$\dot{y}_r = -\dot{x}\sin\theta + \dot{y}\cos\theta = 0 \tag{1.43}$$

$\dot{y}_r$ is zero due to the no-slip condition, $\ddot{y}_r$ could be nonzero under certain conditions. The lateral acceleration can be found by differentiating Eq. (1.43)

$$\begin{aligned}
\ddot{y}_r &= -\ddot{x}\sin\theta + \ddot{y}\cos\theta - \dot{\theta}\left(\dot{x}\cos\theta + \dot{y}\sin\theta\right) \\
&= -\ddot{x}\sin\theta + \ddot{y}\cos\theta - \dot{\theta}\dot{x}_r
\end{aligned} \tag{1.44}$$

Equation (1.43) implies that

$$\dot{y} = \dot{x}\tan\theta \tag{1.45}$$

Differentiating the above equation results in

$$\ddot{y} = \ddot{x}\tan\theta + \dot{x}\dot{\theta}(1 + \tan^2\theta) \tag{1.46}$$

Substituting this into Eq. (1.44) yields

$$\ddot{y}_r = \dot{\theta}\left(\dot{x}(1+\tan^2\theta)\cos\theta - \dot{x}_r\right) \tag{1.47}$$

However, we have $\dot{x} = \dot{x}_r\cos\theta - \dot{y}_r\sin\theta$ and $\dot{y}_r = 0$. Using these relations with the above equation shows that the lateral acceleration component of the robot becomes

$$\ddot{y}_r = 0 \tag{1.48}$$

Equation (1.48) allows us to neglect the second dynamic equation and simplify the third dynamic equation in (1.42). Finally, the dynamic equations of motion for the robot reduce to the following

$$\begin{aligned}
&\left(m_r + 2m_w + \frac{2I_{wy}}{R^2}\right)\dot{v} - m_r c\omega^2 = \frac{1}{R}(\tau_r + \tau_l) \\
&\left(I_r + 2I_w + m_w\frac{D^2}{2} + I_{wy}\frac{D^2}{2R^2} + m_r c^2\right)\dot{\omega} = \frac{D}{2R}(\tau_r - \tau_l)
\end{aligned} \tag{1.49}$$

where $v = \dot{x}_r$ and $\omega = \dot{\theta}$

Let us now define the generalized state variable

$$\mathbf{q}_g = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta \\ v \\ \omega \end{bmatrix} \tag{1.50}$$

The complete system (kinematic + dynamic) becomes

$$\dot{\mathbf{q}}_g = \begin{bmatrix} q_4\cos q_3 \\ q_4\sin q_3 \\ q_5 \\ m_r c q_5^2/m \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{1}{mR} & \frac{1}{mR} \\ \frac{D}{2IR} & -\frac{D}{2IR} \end{bmatrix} \begin{bmatrix} \tau_r \\ \tau_l \end{bmatrix} \tag{1.51}$$

where

$$\begin{aligned}
m &= m_r + 2\left(m_w + \frac{I_{wy}}{R^2}\right) \\
I &= I_r + 2I_w + \left(m_w + \frac{I_{wy}}{R^2}\right)\frac{D^2}{2} + m_r c^2
\end{aligned} \tag{1.52}$$

In what follows we will consider that the robot center of gravity is situated on the wheels axis, in this case we have $c = 0$.

1.3.2 Second-order kinematic model

The dynamical model of a nonholonomic system is expressed as (see [19] for details)

$$\begin{aligned} \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{f}(\mathbf{q}, \dot{\mathbf{q}}) &= \mathbf{B}(\mathbf{q})\tau + \mathbf{A}(\mathbf{q})\lambda \\ \mathbf{A}^T(\mathbf{q})\dot{\mathbf{q}} &= 0 \end{aligned} \tag{1.53}$$

where $\mathbf{M}(\mathbf{q})$ is a positive definite symetric inertia matrix, $\mathbf{A}^T(\mathbf{q})$ is the matrix associated with nonholonmic constraints, λ is a vector of Lagrange multipliers and $\mathbf{B}(\mathbf{q})\tau$ is the set of generalized forces applied to the system. As shown in [19] it can be written in state space form as

$$\begin{aligned} \dot{\mathbf{q}} &= \mathbf{G}(\mathbf{q})\mathbf{v} \\ \mathbf{J}(\mathbf{q})\dot{\mathbf{v}} + \mathbf{m}(\mathbf{q}, \mathbf{v}) &= \mathbf{G}^T(\mathbf{q})\mathbf{B}(\mathbf{q})\tau \end{aligned} \tag{1.54}$$

where $v \in \mathbb{R}^m$ is the vector of *pseudo-velocities*, $\mathbf{G}(\mathbf{q})$ is a matrix whose columns are a basis for the null space of $\mathbf{A}^T(\mathbf{q})$, so that $\mathbf{A}^T(\mathbf{q})\mathbf{G}(\mathbf{q}) = \mathbf{0}$ and we have

$$\begin{aligned} \mathbf{J}(\mathbf{q}) &= \mathbf{G}^T(\mathbf{q})\mathbf{M}(\mathbf{q})\mathbf{G}(\mathbf{q}) \\ \mathbf{m}(\mathbf{q}, \mathbf{v}) &= \mathbf{G}^T(\mathbf{q})\mathbf{M}(\mathbf{q})\dot{\mathbf{G}}(\mathbf{q}) + \mathbf{G}^T(\mathbf{q})\mathbf{f}(\mathbf{q}, \dot{\mathbf{q}}) \end{aligned} \tag{1.55}$$

Under the assumption that $\det(\mathbf{G}^T(\mathbf{q})\mathbf{B}(\mathbf{q})) \neq 0$, it is possible to perform a partial linearization via feedback on (1.54) by letting

$$\tau = \left(\mathbf{G}^T(\mathbf{q})\mathbf{B}(\mathbf{q})\right)^{-1}\left(\mathbf{J}(\mathbf{q})\mathbf{u} + \mathbf{m}(\mathbf{q}, \dot{\mathbf{q}})\right) \tag{1.56}$$

where $\mathbf{u} \in \mathbb{R}^m$ is the *pesudo-acceleration* vector. The resulting system is then

$$\begin{aligned} \dot{\mathbf{q}} &= \mathbf{G}(\mathbf{q})\mathbf{v} \\ \dot{\mathbf{v}} &= \mathbf{u} \end{aligned} \tag{1.57}$$

By defining the state $\mathbf{q}_g = [\mathbf{q}^T \ \ \mathbf{v}^T]^T$, system (1.57) can be expressed as

$$\dot{\mathbf{q}_g} = \begin{bmatrix} \mathbf{G}(\mathbf{q}_g)\mathbf{v} \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ I \end{bmatrix}\mathbf{u} \tag{1.58}$$

which is known as the second-order kinematic model of the constrained mechanical system.

The following two properties of the system (1.58) have been established in [18]

- Nonholonomic system (1.58) is controllable.

- The equilibrium point $x_* = 0$ of the nonholonomic system (1.58) can be made Lagrange stable, but can not be made asymptotically stable by a smooth state feedback.

In the case of mobile robot of type unicycle, the dynamical model (1.51) takes the form

$$\begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{1}{R}\cos\theta & \frac{1}{R}\cos\theta \\ \frac{1}{R}\sin\theta & \frac{1}{R}\sin\theta \\ \frac{D}{2R} & -\frac{D}{2R} \end{bmatrix} \begin{bmatrix} \tau_r \\ \tau_l \end{bmatrix} + \begin{bmatrix} \sin\theta \\ -\cos\theta \\ 0 \end{bmatrix} (-mv\omega) \tag{1.59}$$

$$\begin{bmatrix} \sin\theta & -\cos\theta & 0 \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = 0 \tag{1.60}$$

In that case we have

$$\mathbf{f}(\mathbf{q}, \dot{q}) = \mathbf{0}, \qquad \mathbf{M}(\mathbf{q}) = \begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & I \end{bmatrix}$$

$$\mathbf{B}(\mathbf{q}) = \begin{bmatrix} \frac{1}{R}\cos\theta & \frac{1}{R}\cos\theta \\ \frac{1}{R}\sin\theta & \frac{1}{R}\sin\theta \\ \frac{D}{2R} & -\frac{D}{2R} \end{bmatrix}, \qquad \mathbf{G}(\mathbf{q}) = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix}$$

and we have

$$[\mathbf{G}^T(\mathbf{q})\mathbf{B}(\mathbf{q})]^{-1} = \begin{bmatrix} \frac{R}{2} & \frac{R}{D} \\ \frac{R}{2} & -\frac{R}{D} \end{bmatrix}$$

$$\mathbf{J}(\mathbf{q}) = \begin{bmatrix} m & 0 \\ 0 & I \end{bmatrix}$$

$$\mathbf{G}^T(\mathbf{q})\mathbf{M}(\mathbf{q})\dot{\mathbf{G}}(\mathbf{q}) = \mathbf{0}$$

$$\mathbf{m}(\mathbf{q}, \mathbf{v}) = \mathbf{0}$$

By using the input transformation

$$\tau = \left(\mathbf{G}^T(\mathbf{q})\mathbf{B}(\mathbf{q})\right)^{-1} \mathbf{J}(\mathbf{q})\mathbf{u} = \begin{bmatrix} \frac{mR}{2} & \frac{IR}{D} \\ \frac{mR}{2} & -\frac{IR}{D} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{1.61}$$

the second-order kinematic model is obtained as

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \omega \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{1.62}$$

Summarizing, in nonholonomic mechanical systems -such as wheeled mobile robots- it is possible to cancel the dynamic effects via nonlinear state feedback, provided that the dynamic parameters are exactly known and the complete state of the system is measured.

Under these assumptions, the control problem can be directly addressed by choosing $\mathbf{v}$ in such a way that the kinematic model

$$\dot{\mathbf{q}} = \mathbf{G}(\mathbf{q})\mathbf{v}$$

behaves as desired. From $\mathbf{v}$, it is possible to derive the actual control inputs at the generalized force level through (1.61). Since $\mathbf{u} = \dot{\mathbf{v}}$, then $\mathbf{v}$ must be differentiable with respect to time.

Chapter 2

Trajectory Planning

The problem of planning a trajectory for a mobile robot can be divided into two subproblems: finding a path and defining a timing law on the path. However, if the mobile robot is subject to nonholonomic constraints, the first of these two subproblems becomes more difficult. In fact, in addition to meeting the boundary conditions (interpolation of the assigned points and continuity of the desired degree) the path must also satisfy the nonholonomic constraints at all points.

2.1 Path and Timing Law

Assume that one wants to plan a trajectory $\mathbf{q}(t)$, for $t \in [t_i, t_f]$, that drives a mobile robot from an initial configuration $\mathbf{q}(t_i) = \mathbf{q}_i$ to a final configuration $\mathbf{q}(t_f) = \mathbf{q}_f$ in the absence of obstacles. The trajectory $q(t)$ can be broken down into a geometric path $\mathbf{q}(s)$, with $d\mathbf{q}(s)/ds \neq 0$ for any value of s, and a timing law $s = s(t)$, with the parameter s varying between $s(t_i) = s_i$ and $s(t_f) = s_f$ in a monotonic fashion, i.e., with $\dot{s}(t) \geq 0$, for $t \in [t_i, t_f]$. A possible choice for s is the arc length along the path; in this case, it would be $s_i = 0$ and $s_f = L$, where L is the length of the path.

The above space-time separation implies that

$$\dot{\mathbf{q}} = \frac{d\mathbf{q}}{dt} = \frac{d\mathbf{q}}{ds}\dot{s}$$

The generalized velocity vector is then obtained as the product of the vector $\frac{d\mathbf{q}}{ds}$, which is directed as the tangent to the path in configuration space, by the scalar $\dot{s}$, that varies its modulus. Note that the vector $\begin{bmatrix} \frac{dx}{ds} & \frac{dy}{ds} \end{bmatrix}^T \in \mathbb{R}^2$ is directed as the tangent to

the Cartesian path, and has unit norm if s is the cartesian arc length. Nonholonomic constraints of form (A.4) can then be rewritten as

$$\mathbf{A}(\mathbf{q})\dot{\mathbf{q}} = \mathbf{A}(\mathbf{q})\frac{d\mathbf{q}}{ds}\dot{s} = 0$$

If $\dot{s}(t) > 0$, for $t \in [t_i, t_f]$, one has

$$\mathbf{A}(\mathbf{q})\frac{d\mathbf{q}}{ds} = 0 \tag{2.1}$$

This condition, that must be verified at all points by the tangent vector on the configuration space path, characterizes the notion of geometric path admissibility induced by kinematic constraint (A.4) that actually affects generalized velocities.

Geometrically admissible paths can be explicitly defined as the solutions of the nonlinear system

$$\frac{d\mathbf{q}}{ds} = \mathbf{G}(\mathbf{q})\tilde{\mathbf{u}} \tag{2.2}$$

where $\tilde{\mathbf{u}}$ is a vector of geometric inputs that are related to the velocity inputs $\mathbf{u}$ by the relationship $\mathbf{u}(t) = \tilde{\mathbf{u}}(s) \cdot s(t)$. Once the geometric inputs $\tilde{\mathbf{u}}(s)$ are assigned for $s \in [s_i, s_f]$, the path of the robot in configuration space is uniquely determined. The choice of a timing law $s = s(t)$, for $t \in [t_i, t_f]$, will then identify a particular trajectory along this path.

In the case of a mobile robot with unicycle-like kinematics, constraint (1.6) gives the following condition for geometric admissibility of the path

$$\begin{bmatrix} \sin\theta & -\cos\theta & 0 \end{bmatrix} \frac{d\mathbf{q}}{ds} = \frac{dx}{ds}\sin\theta - \frac{dy}{ds}\cos\theta = 0$$

This condition implies that the tangent to the Cartesian path must be aligned with the robot sagittal axis. Geometrically admissible paths for the unicycle are the solutions of the system

$$\begin{aligned} \frac{dx}{ds} &= \tilde{v}\cos\theta \\ \frac{dy}{ds} &= \tilde{v}\sin\theta \\ \frac{d\theta}{ds} &= \tilde{\omega} \end{aligned} \tag{2.3}$$

where $\tilde{v}$, $\tilde{\omega}$ are related to v, ω by

$$\begin{aligned} v(t) &= \tilde{v}(s)\dot{s}(t) \\ \omega(t) &= \tilde{\omega}(s)\dot{s}(t) \end{aligned} \tag{2.4}$$

2.2 Path Planning

The property of flat outputs of the unicycle allows to solve the planning problem efficiently. In fact, one may use any interpolation scheme to plan the path of $\mathbf{y}$ such as to satisfy the appropriate boundary conditions. The evolution of the other configuration variables, together with the associated control inputs, can then be computed algebraically from y(s). The resulting configuration space path will automatically satisfy nonholonomic constraints (2.1).

Let's consider the problem of planning a path that drives a unicycle from an initial configuration $\mathbf{q}(s_i) = \mathbf{q}_i = [x_i\ y_i\ \theta_i]^T$ to a final configuration $\mathbf{q}(s_f) = \mathbf{q}_f = [x_f\ y_f\ \theta_f]^T$.

2.2.1 Planning via Cubic polynomials

The problem of path planning can be solved using cubic polynomials which have two desired features. First, they are paths of minimal curvature. And second, they are easy to generate online. A cubic polynomial has 4 coefficients, and hence may be used to satisfy both position and velocity constraints at the initial and final positions. By interpolating the initial values x_i, y_i and the final values x_f, y_f of the flat outputs x, y, and letting $s_i = 0$ and $s_f = 1$, one may use the following cubic polynomials

$$\begin{aligned} x(s) &= s^3 x_f - (s-1)^3 x_i + \alpha_x s^2(s-1) + \beta_x (s-1)^2 \\ y(s) &= s^3 y_f - (s-1)^3 y_i + \alpha_y s^2(s-1) + \beta_y (s-1)^2 \end{aligned} \tag{2.5}$$

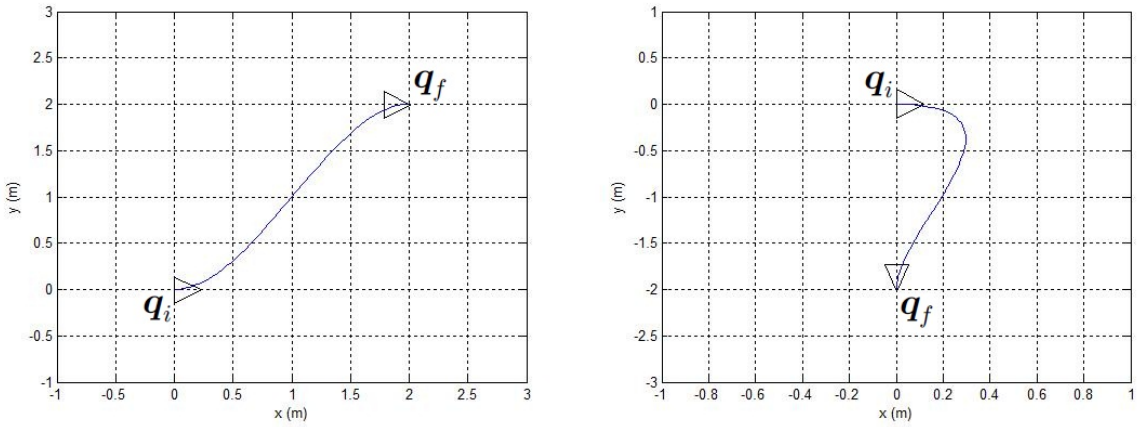
These polynomials satisfy the boundary conditions on x, y. The orientation at each point being related to $\frac{dx}{ds}$, $\frac{dy}{ds}$ by

$$\theta(s) = \operatorname{atan2}(\frac{dy}{ds}(s), \frac{dx}{ds}(s)) + k\pi \qquad k = 0, 1. \tag{2.6}$$

In order to determine the constants α_x, α_y, β_x and β_y it is necessary to impose the additional boundary conditions

$$\begin{aligned} \frac{dx}{ds}(0) &= k_i \cos\theta_i \quad \frac{dx}{ds}(1) = k_f \cos\theta_f \\ \frac{dy}{ds}(0) &= k_i \sin\theta_i \quad \frac{dy}{ds}(1) = k_f \sin\theta_f \end{aligned}$$

where $k_i \neq 0$, $k_f \neq 0$ are free parameters that must however have the same sign. This condition is necessary to guarantee that the unicycle arrives in q_f with the same kind of motion (forward or backward) with which it leaves q_i.



(a) k=2 **(b)** k=2

Figure 2.1: Two parking maneuvers planned via cubic Cartesian polynomials

For example, by letting $k_i = k_f = k > 0$, one obtains

$$\alpha_x = k\cos\theta_f - 3x_f \quad \alpha_y = k\sin\theta_f - 3y_f$$
$$\beta_x = k\cos\theta_i + 3x_i \quad \beta_y = k\sin\theta_i + 3y_i$$

Figures (2.1), (2.2) and (2.3) show the paths produced by the planner that uses cubic polynomials for solving different maneuver problems with different values of k.

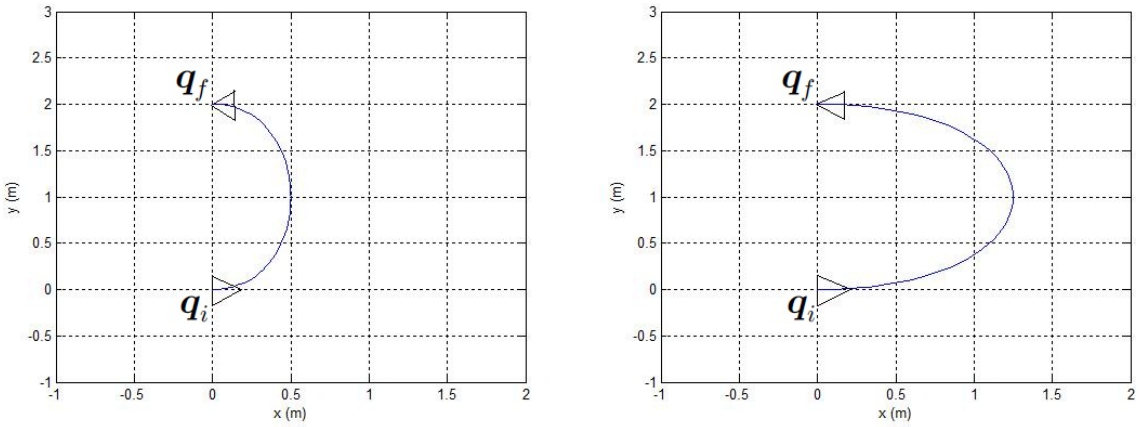
2.2.2 Planning via the chained form

In order to plan a path for the system in chained form, it is first necessary to compute the initial and final values $\mathbf{z}_i$ and $\mathbf{z}_f$ that correspond to $\mathbf{q}_i$ and $\mathbf{q}_f$, by using the change of coordinates (1.28). It is then sufficient to interpolate the initial and final values of z_1 and z_3 (the flat outputs) with the appropriate boundary conditions on the remaining variable $z_2 = \frac{dz_3}{ds} / \frac{dz_1}{ds}$.

Let's adopt the same cubic polynomial approach, under the assumption $z_{1,i} \neq z_{1,f}$, we have the path

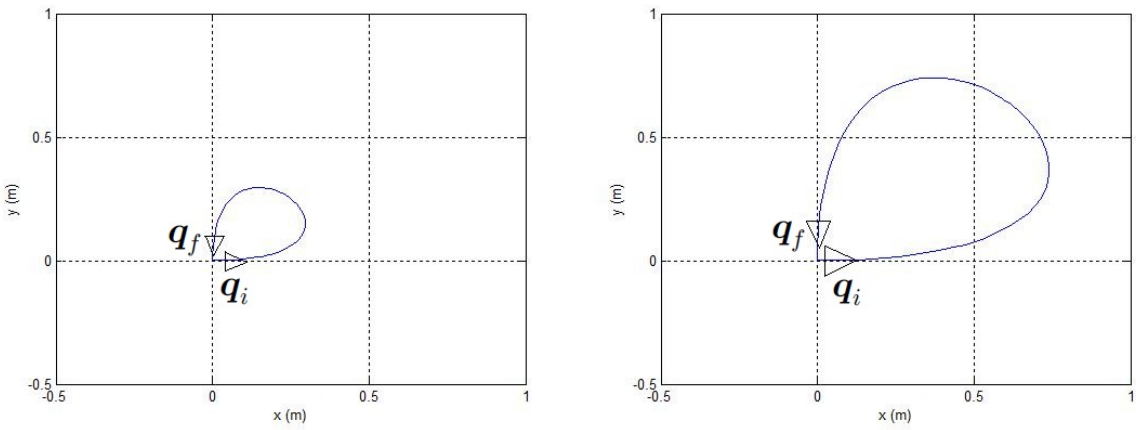
$$\begin{aligned} z_1(s) &= z_{1,f}s - (s-1)z_{1,i} \\ z_3(s) &= s^3 z_{3,f} - (s-1)^3 z_{3,i} + \alpha_3 s^2(s-1) + \beta_3 s(s-1)^2 \end{aligned} \tag{2.7}$$

with $s \in [0, 1]$. The constants α_3 and β_3 can be determined by imposing the boundary conditions on z_2:



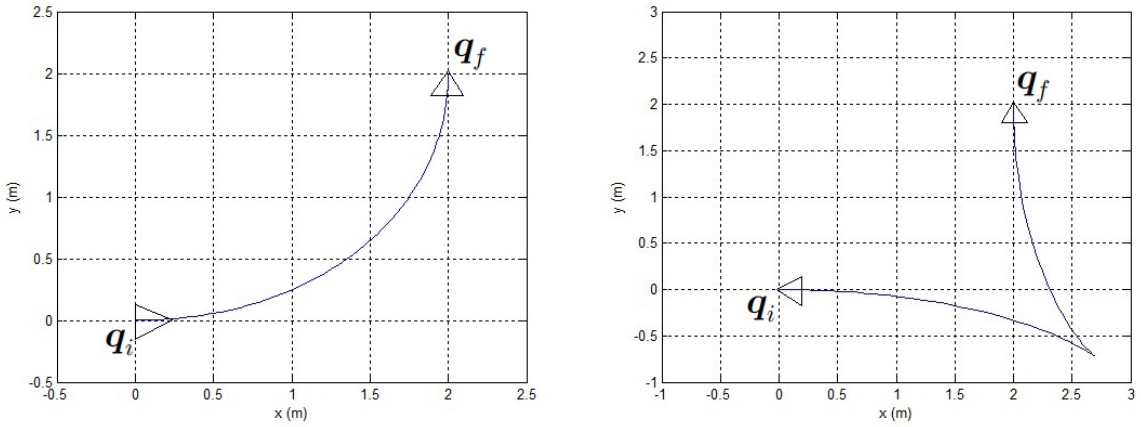
(a) k=2 **(b)** k=5

Figure 2.2: Planning a parallel parking maneuver via cubic Cartesian polynomials



(a) k=2 **(b)** k=5

Figure 2.3: Planning a pure reorientation maneuver via cubic Cartesian polynomials



(a) **(b)**

Figure 2.4: Two parking maneuvers planned via the chained form

$$z_{2,i} = \frac{\frac{dz_3}{ds}(0)}{\frac{dz_1}{ds}(0)}$$
$$z_{2,f} = \frac{\frac{dz_3}{ds}(1)}{\frac{dz_1}{ds}(1)}$$

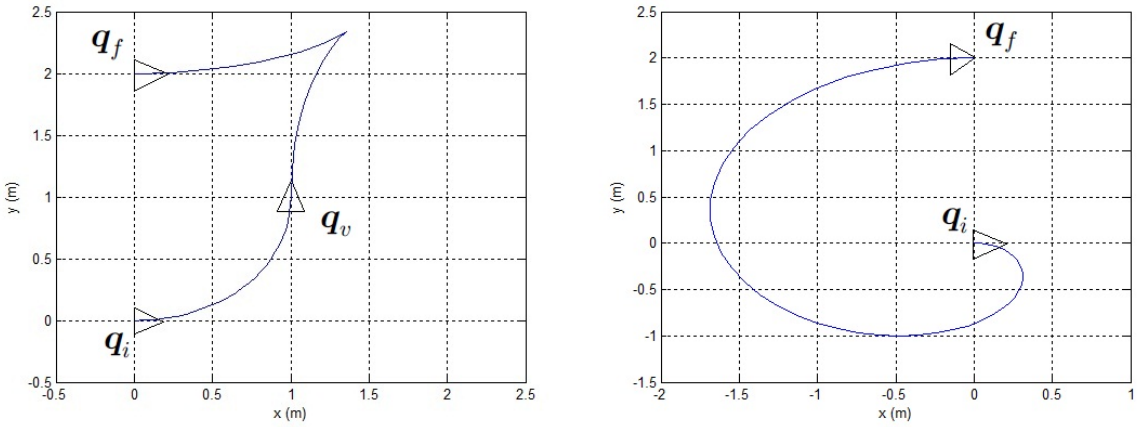
from which

$$\alpha_3 = z_{2,f}(z_{1,f} - z_{1,i}) - 3z_{3,f}$$
$$\beta_3 = z_{2,i}(z_{1,f} - z_{1,i}) + 3z_{3,i}$$

It is obvious that we have a singularity when $z_{1,i} = z_{1,f}$, i.e., when $\theta_i = \theta_f$. One way to avoid this case, is by introducing a via point $\mathbf{q}_v = [x_v \; y_v \; \theta_v]^T$ such that $\theta_v \neq \theta_i$, and solve the original planning problem using two consecutive paths, the first from $\mathbf{q}_i$ to $\mathbf{q}_v$ and the second from $\mathbf{q}_v$ to $\mathbf{q}_f$. Another possibility is to let $z_{1,f} = z_{1,i} + 2\pi$; this corresponds to the same final configuration of the unicycle, but the robot will reach its destination while performing a complete rotation of orientation along the path.

Once the path has been planned for the chained form, the path $\mathbf{q}(s)$ in the original coordinates and the associated geometric inputs $\mathbf{u}(s)$ are reconstructed by inverting the change of coordinates (1.28) and of inputs (1.29), respectively.

The paths produced by this planner for solving different maneuver problems are shown in Figs. 2.4, 2.5.



(a) using via point **(b)** $\theta_f = \theta_f - 2\pi$

Figure 2.5: Planning parallel parking maneuvers via the chained form

2.3 Path planning in presence of obstacles

A complete path planning algorithm should guarantee that the robot reach its target, which makes obstacle avoidance an important issue for the navigation of autonomous mobile robots. In practice, an autonomous robot cannot base its motion planning on complete 'a priori' knowledge of the environment. The robot must rather use its sensors to perceive the environment and plan accordingly. In other words, the robot must incorporate new information gained during plan execution. As time goes on, the environment changes and the robot's sensors gather new information.

Obstacle avoidance adds a second level of difficulty. At this level we should take into account both the constraints due to the obstacles (i.e., dealing with the configuration parameters of the system) and the nonholonomic constraints linking the parameter derivatives. It appears necessary to combine geometric techniques addressing the obstacle avoidance together with control theory techniques addressing the special structure of the nonholonomic motions.

The Bug algorithms, termed **Bug1** and **Bug2** [66], provide a way to overcome unexpected obstacles in the robot motion from a start point $\mathbf{q}_{start}$, to a goal point $\mathbf{q}_{goal}$. The goal of the algorithms is to generate a collision free path from the $\mathbf{q}_{start}$ to $\mathbf{q}_{goal}$ with the underlying principle based on driving around the detected obstacles. These

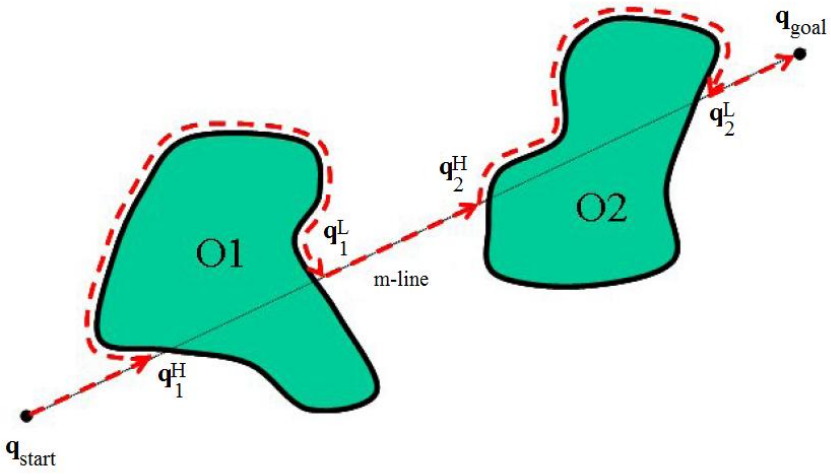


Figure 2.6: Obstacle avoidance with Bug 2 algorithm.

algorithms consist of two reactive modes of motion, termed *behaviors*, and transition conditions for switching between them. The two behaviors are moving directly toward the target and following an obstacle boundary. When the robot hits an obstacle it switches from moving toward the target to boundary following. It leaves the obstacle boundary when a leaving condition, which ensures that the distance to the target decreases, holds.

Fig. 2.6 represents the path generated by Bug 2 for two obstacles.

The Bug approach minimizes the computational burden on the robot while still guaranteeing global convergence to the target. However, the Bug algorithms do not make the best use of the available sensory data to produce short paths. These algorithms use mainly contact sensors. To mend this, a modification to this algorithm termed **TangentBug** was proposed in [56]. TangentBug exploits range data. It constructs a local tangent graph, or LTG (introduced in [65]), and uses it to produce paths which often resemble the shortest path to the goal (see [56] for details).

The steps of the TangentBug algorithm are:

- Move towards $\mathbf{q}_{goal}$ along the locally optimal direction on the current LTG subgraph, until one of the following events occurs:
  - $\mathbf{q}_{goal}$ is reached. Stop.

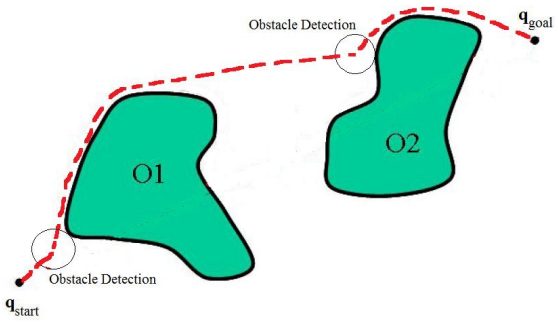


Figure 2.7: Obstacle avoidance with TangentBug algorithm using sensors of range R

  - A local minimum of $d(x, \mathbf{q}_{goal})$ is detected. Go to step 2.

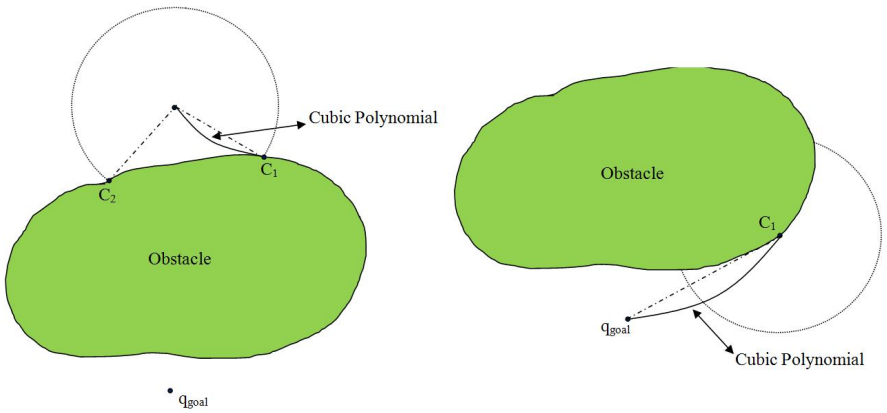
- Choose a boundary following direction. Move around the boundary using the LTG while recording $d_{followed}(\mathbf{q}_{goal})$, the minimal distance along the followed obstacle's boundary to $\mathbf{q}_{goal}$ and $d_{reach}(\mathbf{q}_{goal})$, the minimal distance within the visible environment to $\mathbf{q}_{goal}$, until one of the following occurs:

  - $\mathbf{q}_{goal}$ is reached. Stop.
  - The leaving condition $d_{reach}(\mathbf{q}_{goal}) < d_{followed}(\mathbf{q}_{goal})$ holds. Go to Step 1.
  - The robot completes a loop around the obstacle. $\mathbf{q}_{goal}$ is unreachable. Stop.

Fig. 2.7 represents the path generated by TangantBug for two obstacles.

The problem of this algorithm is that the robot moves suddenly when it switches between the two behaviors (moving toward the target and following an obstacle boundary). This sudden change in path is not desirable for some trajectory tracking control schemes, which require a smooth path with continuous speed and curvature and bounded accelerations.

To overcome this shortcoming, we modified the algorithm in such a way that avoids such abrupt transitions. To this end, cubic polynomials (2.5) are generated to connect the robot and the target in *approach-target* behavior, and the robot and the obstacle in *approach-obstacle* behavior. Constants α_x, α_y, β_x and β_y are adjusted in such a way as to guarantee continuous velocity and curvature, as shown in Fig. 2.8.



(a) **(b)**

Figure 2.8: Modified TangentBug algorithm for path planning: (a) cubic polynomial is generated to connect the robot's current position and an LTG node C1; (b) a cubic polynomial is generated to connect the current position and the target $\mathbf{q}_{goal}$.

2.4 Trajectory Planning

Trajectory planning process consists of choosing a timing law $s = s(t)$ for a certain path $\mathbf{q}(s), s \in [s_i, s_f]$. However, actuator limitations should be respected. For example, in the case of differential-drive unicycle the wheel angular speeds ω_r and ω_l are subject to bounds of the form

$$\begin{aligned} |\omega_r(t)| &\leq \omega_{rmax}, \qquad \forall t \\ |\omega_l(t)| &\leq \omega_{lmax}, \qquad \forall t \end{aligned}$$

Through Eq. (1.5), these bounds can be mapped to constraints on v and ω.

$$\begin{aligned} |v(t)| &\leq v_{max}, \qquad \forall t \\ |\omega(t)| &\leq \omega_{max}, \qquad \forall t \end{aligned} \tag{2.8}$$

So it is necessary to verify whether the velocities along the planned trajectory are admissible. However, it is possible to slow down the timing law via uniform scaling in the case where velocities are inadmissible. To this end, it is convenient to rewrite the timing law by replacing t with the normalized time variable $\tau = t/T$, with $T = t_f - t_i$. From (2.4), we have

$$\begin{aligned} v(t) &= \tilde{v}(s)\frac{ds}{d\tau}\frac{d\tau}{dt} = \tilde{v}(s)\frac{ds}{d\tau}\frac{1}{T} \\ \omega(t) &= \tilde{\omega}(s)\frac{ds}{d\tau}\frac{d\tau}{dt} = \tilde{\omega}(s)\frac{ds}{d\tau}\frac{1}{T} \end{aligned} \tag{2.9}$$

and therefore is sufficient to increase T (i.e., the duration of the trajectory) to reduce uniformly v and ω, so as to stay within the given bounds.

It is also possible to plan directly a trajectory without separating the geometric path from the timing law. To this end, all the techniques presented before can be used with the time variable t directly in place of the path parameter s. A drawback of this approach is that the duration $t_f - t_i$ of the trajectory is fixed, and uniform scaling cannot be used to satisfy bounds on the velocity inputs. In fact, an increase (or decrease) of $t_f - t_i$ would modify the geometric path associated with the planned trajectory.

Once the timing law of $x(t)$ and $y(t)$ have been determined, timing laws of θ, v and ω are then derived using kinematic equations (1.3)

$$\begin{aligned} \theta(t) &= \text{atan2}(\dot{y}(t), \dot{x}(t)) \\ v(t) &= \sqrt{(\dot{x}(t))^2 + (\dot{y}(t))^2} \\ \omega(t) &= \frac{\dot{x}(t)\cdot\ddot{y}(t) - \ddot{x}(t)\cdot\dot{y}(t)}{(\dot{x}(t))^2 + (\dot{y}(t))^2} \end{aligned} \tag{2.10}$$

A possible choice of s is $s(t) = 0.5\tanh(\sigma(t - T/2)) + 0.5$, with $\sigma > 0$. This function has small initial and final velocities, which is very suitable for practical applications.

Chapter 3

Motion Control

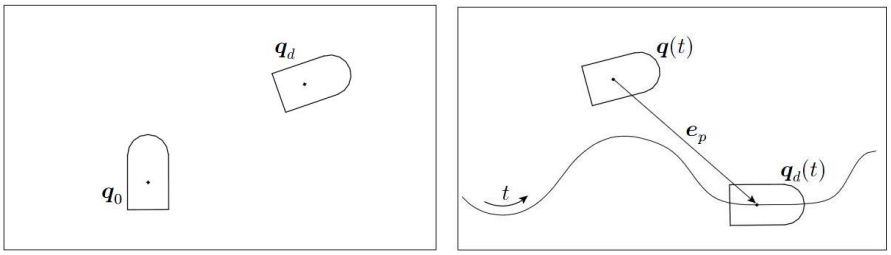
3.1 Introduction

From the theoretical point of view, the control of nonholonomic systems presents interesting aspects. First of all, the control problem is a nonlinear one since a local linearization of the system is not controllable. Moreover, controllability in the nonlinear setting — which is strictly related to the nonholonomic nature of the system— does not imply stabilizability by smooth time-invariant feedback [17].

In this chapter, a unicycle-like vehicle is considered, although some of the presented control schemes may be extended to other kinds of mobile robots. Two basic control problems, illustrated in Fig. 3.1, will be considered [92]:

- *Posture stabilization*: the robot must asymptotically reach a given posture, i.e., a desired configuration $\mathbf{q}_d$, starting from an initial one $\mathbf{q}_0$.
- *Trajectory tracking*: the robot must asymptotically track a desired Cartesian trajectory $(x_d(t), y_d(t))$, starting from an initial configuration $q_0 = [x_0 \ y_0 \ \theta_0]^T$ that may or may not be 'matched' with the trajectory.

We will discuss Integral Sliding Mode controller combined with feedback linearization. In addition, we will study the method known as *Inversion and Immersion* developed by A. Astolfi and R. Ortega [8] to assess the degree of applicability and performance on a system of nonholonomic robot in relation to more usual methods in nonlinear control theory. Simulations will be performed to assess the validity and quality of both control techniques.



(a) Posture stabilization **(b)** Trajectory tracking

Figure 3.1: Control problems for a unicycle

3.2 Feedback Linearization

3.2.1 Input/output linearization

Let's consider the nonholonomic system described in section 1.3.2

$$\begin{aligned} \dot{\mathbf{q}} &= \mathbf{G}(\mathbf{q})\mathbf{v} \\ \dot{\mathbf{v}} &= \mathbf{u} \end{aligned} \tag{3.1}$$

where $\mathbf{q} \in \mathbb{R}^n$ and $\mathbf{v} \in \mathbb{R}^m$, and this system is subject to $n - m$ nonholonomic constraints. As shown in [19], for a nonholonomic system with n degrees of freedom and m actuators, there exists an output vector function $\mathbf{y} = h(\mathbf{q})$ and a static state feedback control $\mathbf{u}(\mathbf{q}, \mathbf{v})$ such that the closed loop is stable, and the output $\mathbf{y} = h(\mathbf{q})$ asymptotically converges to zero. This can be achieved by feedback linearization.

We start by choosing the output function

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} \tag{3.2}$$

which depends on the configuration state variable $\mathbf{q}$ only, but not on the state $\mathbf{v}$, such that the largest linearizable subsystem is obtained by differentiating this output function as follows

$$\begin{aligned} \dot{y} &= \nabla_{\mathbf{q}} h(\mathbf{q})\dot{\mathbf{q}} \\ &= \nabla_{\mathbf{q}} h(\mathbf{q})\mathbf{G}(\mathbf{q})\mathbf{v} \end{aligned} \tag{3.3}$$

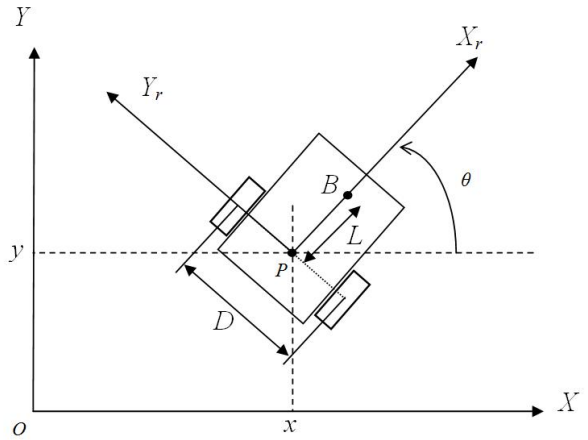


Figure 3.2: Unicycle mobile robot.

By differentiating again, one may write

$$\ddot{y} = \mathbf{F}(\mathbf{q}, \mathbf{v}) + \mathbf{D}(\mathbf{q})\mathbf{u} \tag{3.4}$$

where

$$\mathbf{F}(\mathbf{q}, \mathbf{v}) = \frac{\partial}{\partial \mathbf{q}} \left[\nabla_{\mathbf{q}} h(\mathbf{q})\mathbf{G}(\mathbf{q})\mathbf{v}\right] \mathbf{G}(\mathbf{q})\mathbf{v} \tag{3.5}$$

$$\mathbf{D}(\mathbf{q}) = \nabla_{\mathbf{q}} h(\mathbf{q})\mathbf{G}(\mathbf{q}) \tag{3.6}$$

By choosing $h(\mathbf{q})$ in such a way that the matrix $\mathbf{D}(\mathbf{q})$ is nonsingular for all $\mathbf{q}$, then linearization is achieved by the following feedback control

$$\mathbf{u} = \mathbf{D}^{-1}(\mathbf{q})(\mathbf{z} - \mathbf{F}(\mathbf{q}, \mathbf{v})) \tag{3.7}$$

where $\mathbf{z} \in \mathbb{R}^m$ is the new external control input. And the resulting system is

$$\ddot{\mathbf{y}} = \mathbf{z} \tag{3.8}$$

In the case of unicycle (1.62), input-output linearizability is guaranteed through this choice of output function

$$\mathbf{y} = h(\mathbf{q}) = \begin{bmatrix} x + L\cos\theta \\ y + L\sin\theta \end{bmatrix} \tag{3.9}$$

with $L \neq 0$. They represent the Cartesian coordinates of a point B located along the sagittal axis of the unicycle at a distance $|L|$ from the wheels axis (see Fig. 3.2). This choice of function h results in the following expressions of $\mathbf{D}$ and $\mathbf{F}$

$$\mathbf{D}(\mathbf{q}) = \begin{bmatrix} \cos\theta & -L\sin\theta \\ \sin\theta & L\cos\theta \end{bmatrix} \tag{3.10}$$

$$\mathbf{F}(\mathbf{q}) = \begin{bmatrix} -v\omega\sin\theta - L\omega^2\cos\theta \\ v\omega\cos\theta - L\omega^2\sin\theta \end{bmatrix} \tag{3.11}$$

It is easy to verify that matrix $D(\mathbf{q})$ is nonsingular for all $\mathbf{q}$ given that $L \neq 0$.

Remark 1 *Friction forces that have not been taken into account in the modelling step, in addition to measurement errors, parameter uncertainties and other unmodeled dynamics appear in (3.7) as a vector* $\mathbf{F}'(t) \in \mathbb{R}^{2\times 1}$ *as follows*

$$\boldsymbol{u} = \mathbf{D}^{-1}(\boldsymbol{q})(\boldsymbol{z} - \mathbf{F}(\boldsymbol{q}, \boldsymbol{v}) + \mathbf{F}'(t)) \tag{3.12}$$

which can be considered as input disturbances. Denoting the disturbances as a vector function $z_h(t) \in \mathbb{R}^{2\times 1}$*, a real applied input is represented as follows*

$$z_a = z + z_h(t) \tag{3.13}$$

The choice of the output equations means that the controlled point is not situated on the wheel's axis. Sometimes it may be necessary to control the midpoint of the wheel's axis, in this case we resort to dynamic state feedback linearization.

3.2.2 Dynamic State Feedback Linearization

With reference to a generic driftless nonlinear system

$$\dot{\mathbf{q}} = \mathbf{G}(\mathbf{q})\mathbf{v} \qquad \mathbf{q} \in \mathbb{R}^n, \quad \mathbf{v} \in \mathbb{R}^m \tag{3.14}$$

Let us recall that the problem consists in finding, if possible, a feedback compensator of the form [26]

$$\begin{aligned} \dot{\xi} &= a(\mathbf{q}, \xi) + b(\mathbf{q}, \xi)\mathbf{u} \\ \mathbf{v} &= c(\mathbf{q}, \xi) + d(\mathbf{q}, \xi)\mathbf{u} \end{aligned} \tag{3.15}$$

with state ξ and input $\mathbf{u}$, such that the closed-loop system (3.14) and (3.15) is equivalent, under a state transformation, to a linear system.

Now let's consider the second-order kinematic model described in section 1.3.2

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \omega \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{3.16}$$

where u_1 and u_2 are connected with torques generated by wheels through input transformation (1.61), and we define the linearizing output vector as $\eta = [x\ y]^T$. Differentiating η with respect to time yields

$$\dot{\eta} = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{3.17}$$

We notice that the input $\mathbf{u} = [u_1\ u_2]^T$ does not appear in this first order differentiation, second differentiation yields

$$\ddot{\eta} = \begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} u_1\cos\theta - v\omega\sin\theta \\ u_1\sin\theta + v\omega\cos\theta \end{bmatrix} \tag{3.18}$$

showing that only u_1 affects $\ddot{\eta}$, while the second input u_2 cannot be recovered from this second order differential information. To proceed, we need to add an integrator (whose state is denoted by ξ) on the first input u_1

$$\begin{aligned} \xi &= u_1 \\ \dot{\xi} &= a \end{aligned} \tag{3.19}$$

Differentiating further, we obtain

$$\begin{aligned} x^{(3)} &= \dot{u}_1\cos\theta - 2u_1\omega\sin\theta - vu_2\sin\theta - v\omega^2\cos\theta \\ y^{(3)} &= \dot{u}_1\sin\theta + 2u_1\omega\cos\theta + vu_2\cos\theta - v\omega^2\sin\theta \end{aligned} \tag{3.20}$$

Let's define the new input vector

$$\mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} x^{(3)} \\ y^{(3)} \end{bmatrix} \tag{3.21}$$

Equation (3.20) can be put in the form

$$\begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} = \begin{bmatrix} \cos\theta & -v\sin\theta \\ \sin\theta & v\cos\theta \end{bmatrix} \begin{bmatrix} \dot{u_1} \\ u_2 \end{bmatrix} = \mathbf{H}(v,\theta) \begin{bmatrix} a \\ u_2 \end{bmatrix} \tag{3.22}$$

where

$$\begin{aligned} \mu_1 &= z_1 + 2u_1\omega\sin\theta + v\omega^2\cos\theta \\ \mu_2 &= z_2 - 2u_1\omega\cos\theta + v\omega^2\sin\theta \end{aligned} \tag{3.23}$$

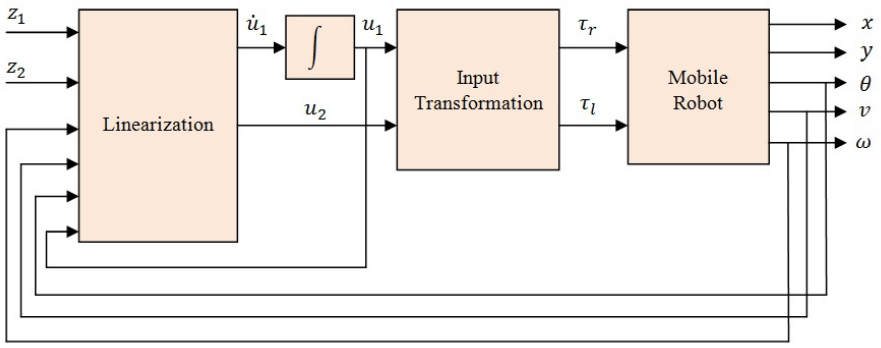


Figure 3.3: System linearization by dynamic state feedback

and the matrix $\mathbf{H}(v, \theta)$ is non singular provided that $v \neq 0$. Under this assumption, we can write

$$\begin{bmatrix} a \\ u_2 \end{bmatrix} = \begin{bmatrix} \cos\theta & -v\sin\theta \\ \sin\theta & v\cos\theta \end{bmatrix}^{-1} \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} \tag{3.24}$$

Then the resulting dynamic compensator is

$$\begin{aligned} a &= \mu_1 \cos\theta + \mu_2 \sin\theta \\ u_2 &= \frac{1}{v}\left(-\mu_1 \sin\theta + \mu_2 \cos\theta\right) \end{aligned} \tag{3.25}$$

substituting (3.23) in (3.25) yields

$$\begin{aligned} a = \dot{u}_1 &= z_1 \cos\theta + z_2 \sin\theta + v\omega^2 \\ u_2 &= \frac{1}{v}\left(-z_1 \sin\theta + z_2 \cos\theta - 2u_1\omega\right) \end{aligned} \tag{3.26}$$

The complete linearizing dynamic feedback block diagram is presented in Fig. (3.3).

Remark 2 *Here the controlled output is the midpoint of the wheels' axis. This, however, comes at the cost that the dynamic compensator (3.26) is not defined when* $v = 0$.

Now, we will consider a robust control strategy based on the method of Integral Sliding Mode.

3.3 Integral Sliding Mode

Sliding mode control has been applied to the trajectory control of robot manipulators [94], [106], and is receiving increasing attention from researches on control of nonholonomic systems with uncertainties. For example, Bloch and Drakunov proposed a sliding

mode control law for the stabilization problem [13], and extended their work to tracking problem [14]. In [43] a sliding mode control was used to guarantee exact tracking of trajectories made by navigation functions. In [105] a sliding mode control law is proposed for asymptotically stabilizing the mobile robot to a desired trajectory, where robot posture was represented using polar coordinates. The benefits of the sliding mode command which makes it very important is its robustness with regard to disturbances and structural uncertainties, i. e. the system response depends on the gradient of the sliding surface and remains insensitive to variations of system parameters and external disturbances. However, during the reaching phase (before Sliding Mode occurs), the system has no such insensitivity property; therefore, insensitivity cannot be ensured throughout an entire response. The robustness during the reaching phase is normally improved by high-gain feedback control. Stability problems that arise inevitably limit the application of such high-gain feedback control schemes.

On the other hand, the concept of Integral Sliding Mode concentrates on the robustness of the motion in the whole state space. The order of the motion equation in this type of Sliding Mode is equal to the dimension of the state space. Therefore, the robustness of the system can be guaranteed throughout an entire response of the system starting from the initial time instance.

For the sake of completeness, in the next section we briefly present the major result of Integral Sliding mode technique presented in [101].

3.3.1 Review of Integral Sliding Mode

For a given dynamic system represented by the following state space equation

$$\dot{x} = f(x) + B(x)u \tag{3.27}$$

where $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$, we suppose that there exists a feedback control law $u = u_0(x)$, such that system (3.27) can be stabilized in a desired way (e.g. its state trajectory follows a reference trajectory with a given accuracy). We denote this ideal closed loop system as

$$\dot{x}_* = f(x) + B(x)u_0 \tag{3.28}$$

where x_* denotes the state trajectory of the ideal system under control u_0. However, systems like (3.27) are normally operating under some uncertainty conditions that may

be generated by parameter variations, unmodeled dynamics and external disturbances etc. Under this consideration a real control system may be summarized with

$$\dot{x} = f(x) + B(x)u + h_d(x,t) \tag{3.29}$$

in which function $h_d(x,t)$ represents the whole perturbation described above and we assume that it is bounded and fulfills the *uncertainty matching condition* (see Remark 1), in other words

$$h_d(x,t) = B(x)u_h \qquad u_h \in \mathbb{R}^m \tag{3.30}$$

For system (3.27), firstly, we design a control like

$$u = u_0 + u_1 \tag{3.31}$$

where u_0 is the ideal control defined in (3.28) and u_1 is designed to be discontinuous for rejecting the perturbation term $h_d(x,t)$. Secondly, we design the *switching function* s as

$$s = s_0(x) + \mu \tag{3.32}$$

with $s, s_0(x), \mu \in \mathbb{R}^m$.

This *switching function* consists of two parts; the first part $s_0(x)$ may be designed as the linear combination of the system states (similar to the conventional Sliding Mode design); and, the second part μ induces the integral term and will be determined below.

To derive the Sliding Mode equation, the time derivative of s on the system trajectories should be made equal to zero; the differential equation $\dot{s} = 0$ should be solved with respect to the control input and the solution u_{eq}, referred to as the *Equivalent Control* should be substituted into the motion equation for u [102].

The control philosophy is to design an integral feedback such that the *Equivalent Control* is

$$u_{1eq} = -u_h \tag{3.33}$$

Condition (3.33) holds if

$$\begin{aligned} \dot{\mu} &= \frac{\partial s_0}{\partial x}\left(f(x) + B(x)u_0\right) \\ \mu(0) &= -s_0(x(0)) \end{aligned} \tag{3.34}$$

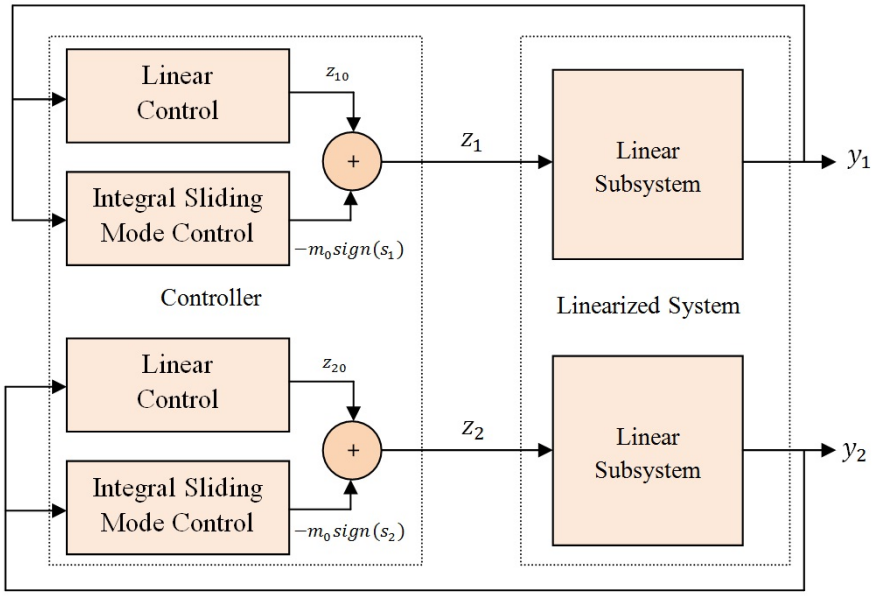


Figure 3.4: Block diagram of controller based on Integral Sliding Mode and feedback linearization to achieve posture stabilization

where $\mu(0)$ is determined based on the requirement $s(0) = 0$ (Sliding Mode occurs starting from the initial time) . The motion equation of the system in Sliding Mode will be the ideal system (3.28).

The Sliding Mode can be enforced using the control

$$u_1 = -M(x)sign(s) \tag{3.35}$$

where $M(x)$ is a positive definite diagonal matrix, under the condition that the matrix $\frac{\partial s_0}{\partial x}B(x)$ is positive definite and the elements of matrix $M(x)$ are large enough.

3.3.2 Posture Stabilization

After feedback linearizing system (3.1), the closed-loop system is equivalent to two linear subsystems of the form

$$\dot{\mathbf{x}}_i = \mathbf{B}_{ii} z_i \qquad i = 1, 2 \tag{3.36}$$

where $\mathbf{x}_i \in \mathbb{R}^{n_i}$, $z_i \in \mathbb{R}$ and $\mathbf{B}_{ii}$ are known matrices. These systems are subject to external disturbances, modelling uncertainties (Feedback linearization requires a

precise knowledge of system parameters) and unmodelled dynamics (motors dynamics and friction forces). Under the assumption that these disturbances fulfill the matching condition, we can write

$$\dot{\mathbf{x}}_i = \mathbf{B}_{ii}\left(z_i + h_{di}(\mathbf{x}, t)\right) \tag{3.37}$$

where $h_i(\mathbf{x}, t)$ is a non-linear perturbation with known upper bound

$$h_{di}(\mathbf{x}, t) \leq |h_{d0}| \qquad \forall t \geq t_0 \tag{3.38}$$

By applying the algorithm of the previous section, we need to design a control z_i as stated in equation (3.31): $z_i = z_{i0} + z_{i1}$, where z_{i0} is predetermined such that system $\mathbf{x}_i = \mathbf{B}_{ii} z_{i0}$ follows a given trajectory with satisfactory accuracy. For example, z_{i0}, may be obtained through linear feedback control, like $z_{i0} = -\mathbf{k}^T \mathbf{x}_i, \mathbf{k} \in \mathbb{R}^{n_i \times 1}$ in which gain vector $\mathbf{k}$ can be determined by Pole Placement or Linear Quadratic Regulator (LQR) methods.

We continue by designing the sliding surface

$$s_i = \quad \mathbf{c}_i^T \mathbf{x}_i + \mu_i \tag{3.39}$$

$$\dot{\mu}_i = \quad -\mathbf{c}_i^T (\mathbf{B}_{ii} z_{i0}) \tag{3.40}$$

$$\mu_i(0) = \quad -\mathbf{c}_i^T \mathbf{x}_i(0) \tag{3.41}$$

in that case the motion equation of the Sliding Mode coincides with that of the ideal system $\mathbf{x}_i = \mathbf{B}_{ii} z_{i0}$, without perturbation. Furthermore, since $s(0) = \mathbf{c}_i^T \mathbf{x} + \mu(0) = 0$, Sliding Mode will occur from the initial time $t = 0$. The second part of the control i.e. μ_{i1} can be designed as following

$$z_{i1} = m_0(\mathbf{x}) sign(s_i) \tag{3.42}$$

where $m_0(\mathbf{x}) \geq |h_0|$.

The overall control block diagram is shown in Fig. 3.4.

3.3.3 Trajectory Tracking

Given a smooth bounded reference trajectory

$$\mathbf{y}_d(t) = h(\mathbf{q}_d(t)) \tag{3.43}$$

which is generated by a trajectory generator which satisfies nonholonomic constraints (1.6), then the tracking control problem is to design a feedback control law for system (3.1) with output equation $\mathbf{y}(t) = h(\mathbf{q}(t))$ such that the tracking error

$$\mathbf{e}(t) = \mathbf{y}(t) - \mathbf{y}_d(t) \tag{3.44}$$

is bounded and asymptotically tends to zero. By differentiating (3.44) twice, one may write using (3.8)

$$\begin{aligned} \ddot{\mathbf{e}} &= \ddot{\mathbf{y}} - \ddot{\mathbf{y}}_d \\ &= \mathbf{z} - \ddot{\mathbf{y}}_d + h_d(\mathbf{e}, t) \end{aligned} \tag{3.45}$$

As in the previous section, we design a control $\mathbf{z} = \mathbf{z}_0 + \mathbf{z}_1$. For the first part $\mathbf{z}_0$ we consider the PID controller

$$\mathbf{z}_0 = -(k_p + \frac{k_i}{p} + k_d p)\mathbf{I}_2 + \ddot{\mathbf{y}}_d \tag{3.46}$$

where p denotes the Laplace transform variable, while $\mathbf{z}_1$ is designed as in previous section to counteract the perturbations $h(\mathbf{e}, t)$ with respect to initial conditions.

3.3.4 Simulation Results

we have performed computer simulations using SIMULINK in order to show the performance and robustness of the proposed controller for both control problems of a wheeled mobile robot which is subjected to parametric and nonparametric uncertainties. The values of WMR parameters are depicted in Table 3.1 and are chosen to match with a real world mobile robot (KOALA Fig. 1.1a).

A. Posture stabilization

We start by applying a linear controller $z = -\mathbf{k}^T\mathbf{y}$ to the undisturbed linearized system. At $t = 0$, the initial posture of the robot is $(x(0), y(0), \theta(0)) = (3.0m, 2.0m, -\pi rad)$ which corresponds to output value $(y_1(0), y_2(0)) = (2.9m, 2.0m)$. $\mathbf{k}$ is derived using LQR technique, thus $[k_1 \;\; k_2]^T = [1.0 \;\; 1.73]^T$. Fig. 3.5 shows stabilization of the system without disturbances about the origin using linear feedback controller, while Fig. 3.6 shows the evolution of θ and linear and angular velocities of the robot. We observe that

Parameter	Description	Value
D	Distance between two wheels	$0.3m$
R	Driving wheels radius	$0.044m$
L	Distance of point B from robot's center of gravity P	$0.1m$
m_r	The mass of the robot without the driving wheels and the rotors of the DC motors	$3.4kg$
m_w	The mass of each driving wheel plus the rotor of its motor	$0.2kg$
I_r	The moment of inertia of the robot without the driving wheels about a vertical axis through P	$1kgm^2$
I_w	The moment of inertia of each wheel about its diameter	$0.00002kgm^2$
I_wy	The moment of inertia of each wheel about its axis	$0.0001kgm^2$

Table 3.1: Model parameters of wheeled mobile robot and their attributed values for the simulations

the LQR controller ensures a good and fast response (within actuators limitations[1].) and it succeeds in driving the point B of the robot to the origin.

A less satisfactory response is obtained by introducing uncertainties to the dynamic parameters like mass and inertia[2] of the robot up to 60% of their original values. The controller, however, succeeds in driving the robot to the origin as shown in Fig. 3.7.

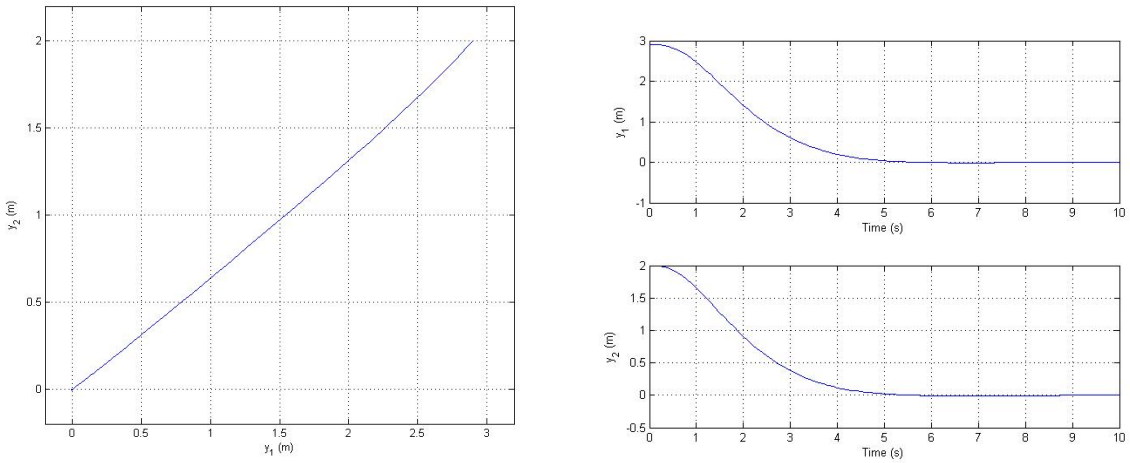
Adding external disturbances to the input in the form of random signals results in an unstable behavior of the closed-loop system, and the controller does not drive the robot to its destination as shown in Fig. 3.8.

Now we add the Sliding Mode based controller without the integral effect, and we use as parameters: $\mathbf{c}^T = [1.0 \;\; 1.0]$ and $m_0 = 5$. Figs. 3.9 shows a better response than that of the LQR controller.

Now we add the Integral Sliding Mode based controller described in Section 3.3.1, and we use as parameters: $\mathbf{c}^T = [0.2 \;\; 1]$ and $m_0 = 5$. Figs. 3.10 through 3.13 show a much better response than that of the LQR controller alone or that of traditional Sliding Mode controller in presence of parametric uncertainties, and the system response is similar to that of the LQR controller without parametric uncertainties.

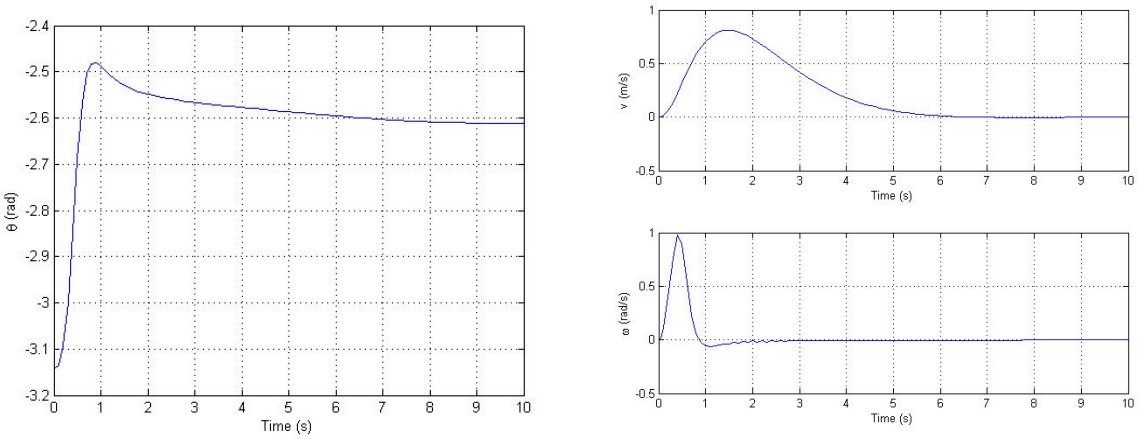
[1] We considered here that $|v|_{max} = 1$ and $|\omega|_{max} = 1$

[2] The kinematic parameters like D and R are geometric and easy to measure, they are reasonably considered to be certain.



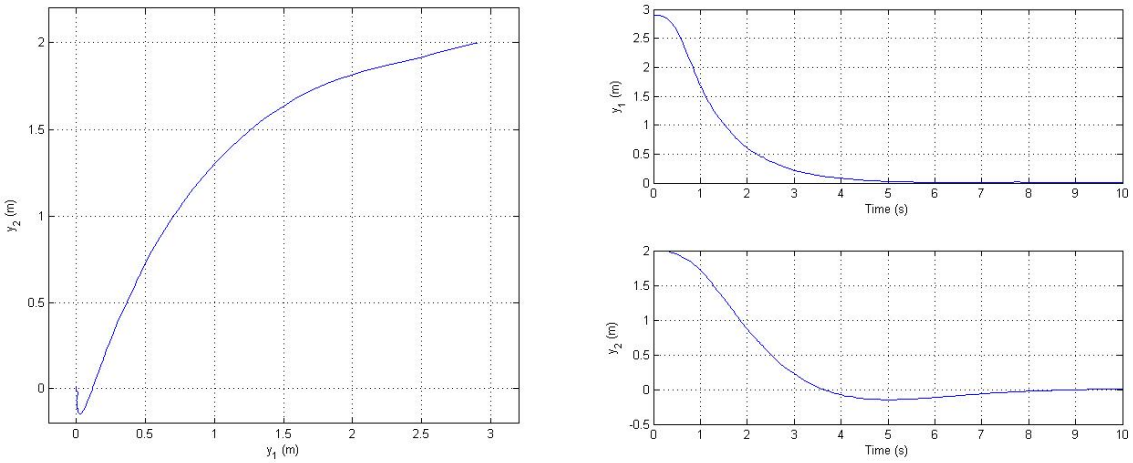
(a) Trajectory **(b)** Time Responses

Figure 3.5: Stabilization of undisturbed robot system using linear controller



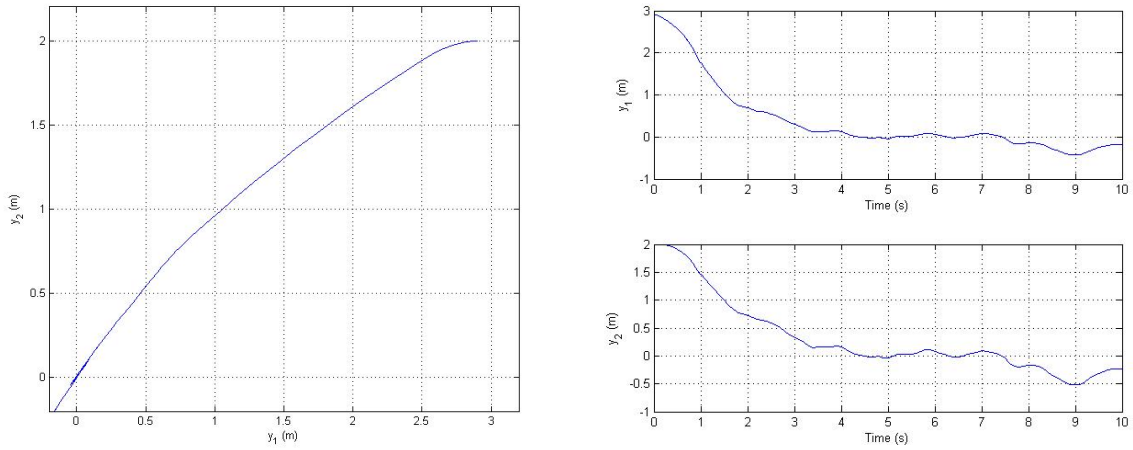
(a) Robot's Orientation **(b)** Velocities

Figure 3.6: Robot's orientation and linear and angular velocities



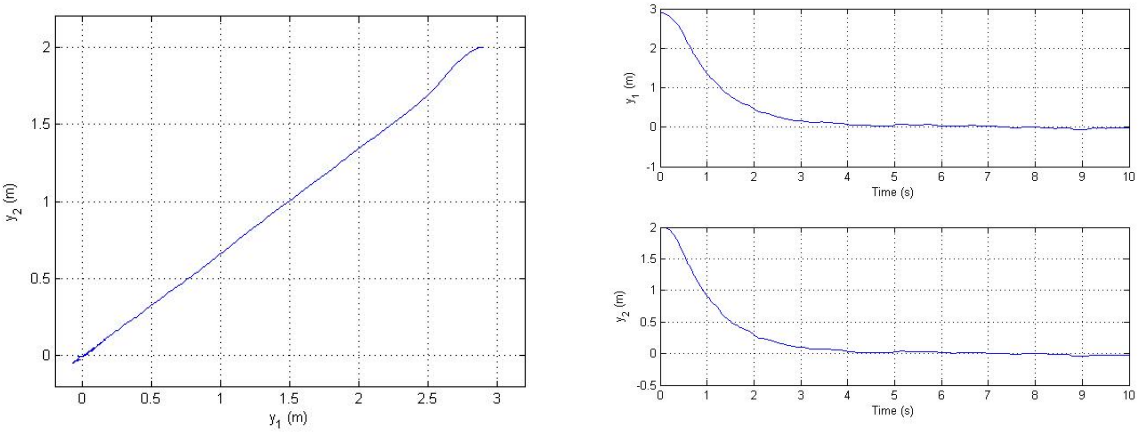
(a) Trajectory **(b)** Time Responses

Figure 3.7: Stabilization of robot system using linear controller in presence of parametric uncertainties.



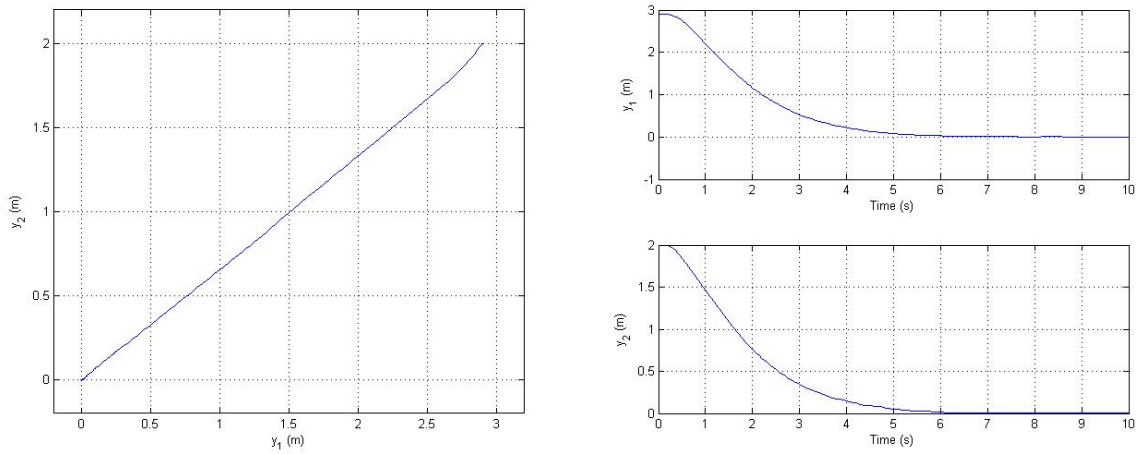
(a) Trajectory **(b)** Time Responses

Figure 3.8: Stabilization of robot system using linear controller in presence of external disturbances.



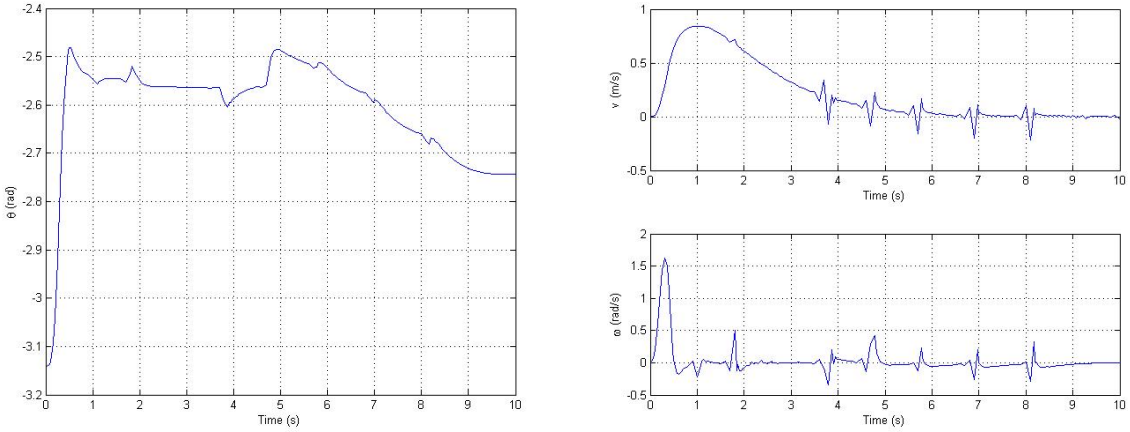
(a) Trajectory **(b)** Time Responses

Figure 3.9: Stabilization of robot system using traditional Sliding Mode controller in presence of external disturbances.



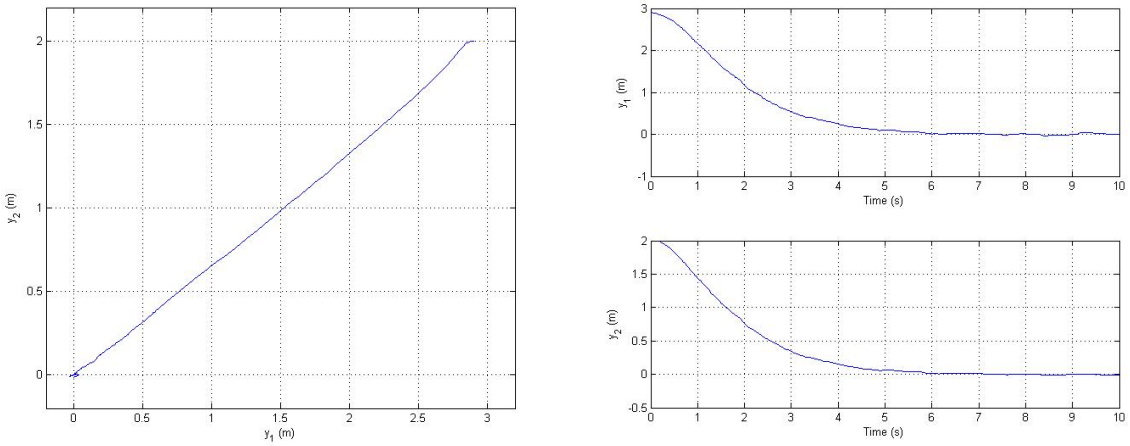
(a) Trajectory **(b)** Time Responses

Figure 3.10: Stabilization of robot system using linear controller plus Integral Sliding Mode controller in presence of parametric uncertainties.



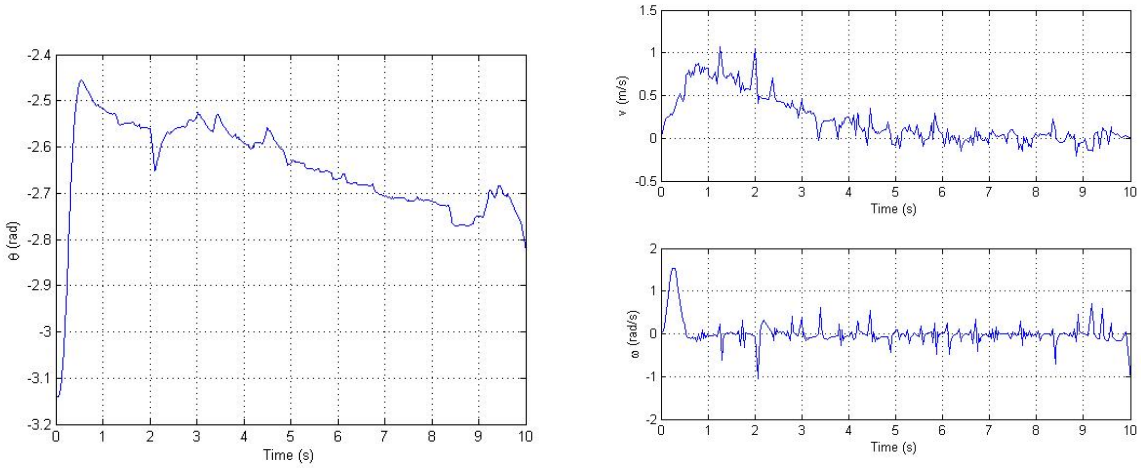
(a) Robot's Orientation **(b)** Velocities

Figure 3.11: Robot's orientation and linear and angular velocities



(a) Trajectory **(b)** Time Responses

Figure 3.12: Stabilization of robot system using linear controller plus Integral Sliding Mode controller in presence of external disturbances.



(a) Robot's Orientation **(b)** Velocities

Figure 3.13: Robot's orientation and linear and angular velocities

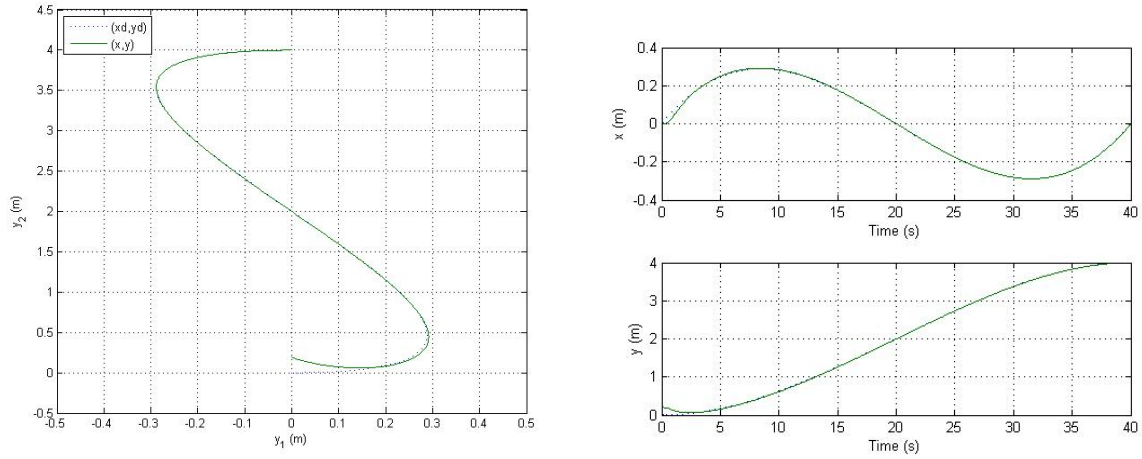
We then apply external disturbances to the system with Integral Sliding Mode controller. Fig. 3.12 shows that the system response is more similar to that of the LQR controller in the absence of external disturbances, and Fig. 3.13b shows that linear and angular velocities are within accepted limits.

B. Trajectory Tracking

We start by applying the linear controller (3.46) to the undisturbed linearized system in order to track a trajectory generated using a cubic polynomials as (discussed in chapter 2) to perform a parallel parking maneuver from point (0,0) to point (0,4). The initial position of the robot is $(y_1(0), y_1(0)) = (0m, 0.1m)$. We chose $k_p = 9.17$,$k_i = 0.72$ and $k_d = -10.59$. Figs. 3.14 through 3.16 show a good tracking, and inputs values are within physical limits.

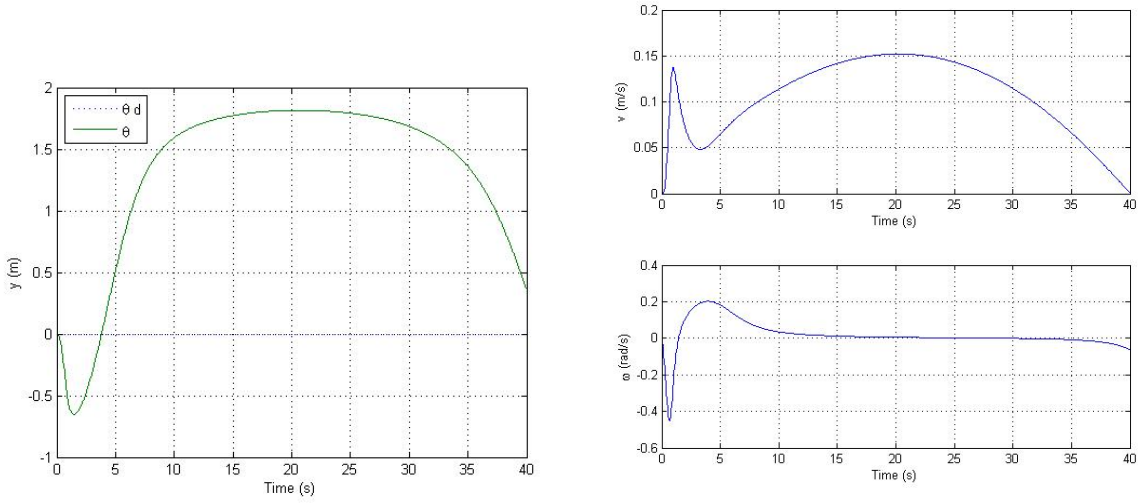
Adding external disturbances to the input in the form of random signals in addition to parameters uncertainties results in a poor tracking as we can see in Fig. 3.17.

Now we add the Integral Sliding Mode based controller described in Section 3.3.1, and we use as parameters: $\mathbf{c}^T = [0.2 \quad 1]$ and $m_0 = 5$. Figs. 3.18 through 3.20 show a much better response than that of the PID controller alone in presence of parametric



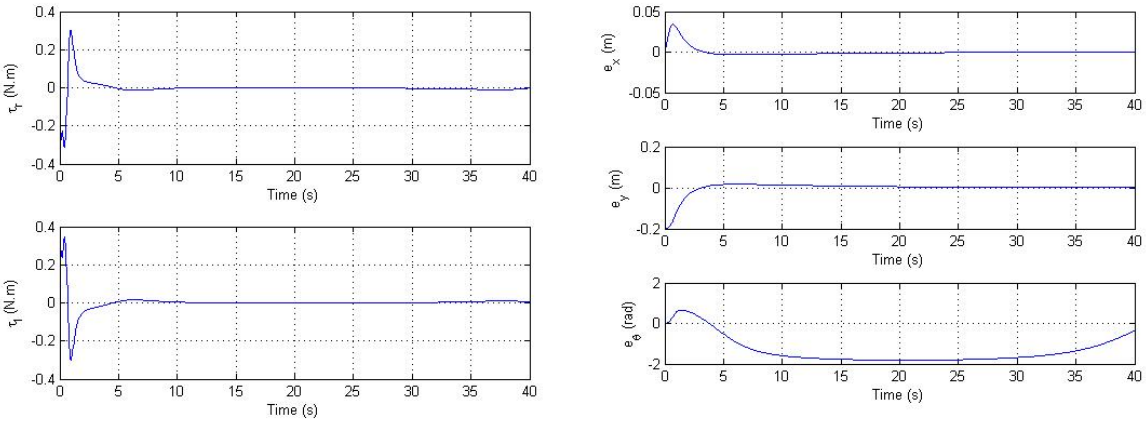
(a) Trajectory **(b)** Time Responses

Figure 3.14: Trajectory tracking of robot system using PID controller in the absence of disturbances.



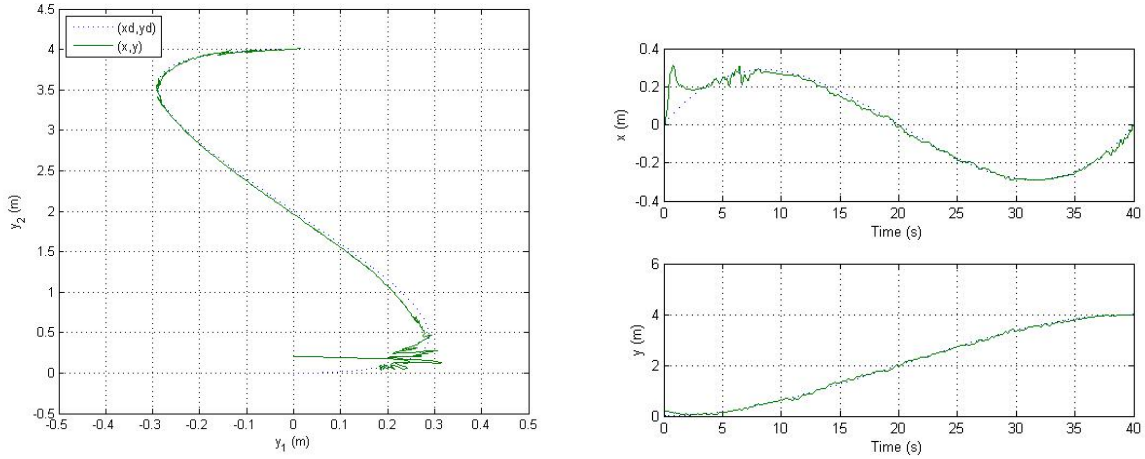
(a) Robot's Orientation **(b)** Velocities

Figure 3.15: Robot's orientation and linear and angular velocities



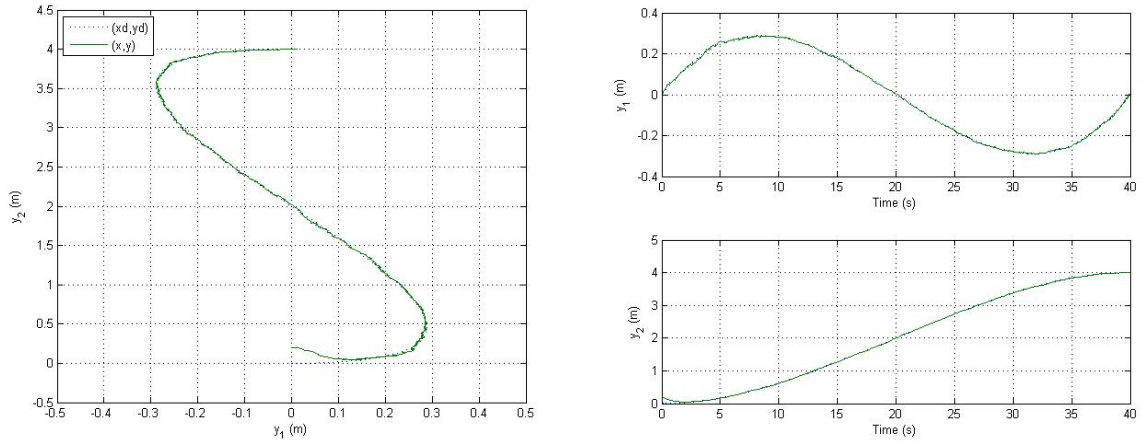
(a) Right and Left Torques **(b)** Tracking errors

Figure 3.16: Applied torques and tracking errors.



(a) Trajectory **(b)** Time Responses

Figure 3.17: Trajectory tracking of robot system using PID controller in the presence of disturbances.



(a) Trajectory **(b)** Time Responses

Figure 3.18: Trajectory tracking of robot system using PID controller plus Integral Sliding Mode controller in presence of external disturbances and parametric uncertainties.

uncertainties and external disturbances, and the system response is similar to that of the PID controller without parametric uncertainties and even better.

Table 3.2 summarizes the simulation results. Basically, the performance of Integral Sliding Mode controller is very simillar to that of a linear controller in absence of disturbances.

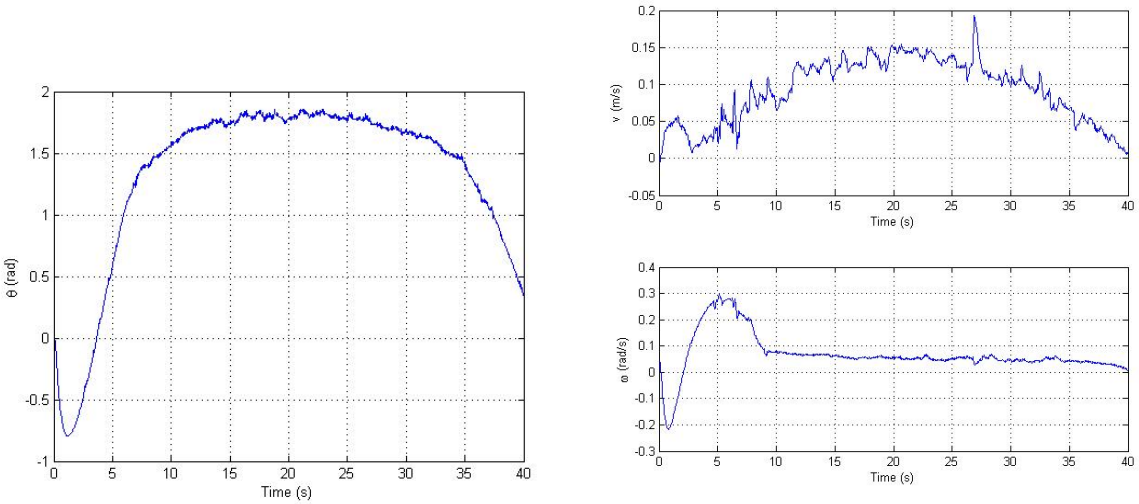
Control Method	Response Time	Precision	Disturbance Rejection
Linear Controller	Good	Good	Low
Integral Sliding Mode	Good	Good	High

Table 3.2: Comparison of performance between linear controller and Integral Sliding Mode controller

Remark 3 *This first approach to motion control is based on the output error rather than state error. Note that the orientation, whose evolution is governed by the equation*

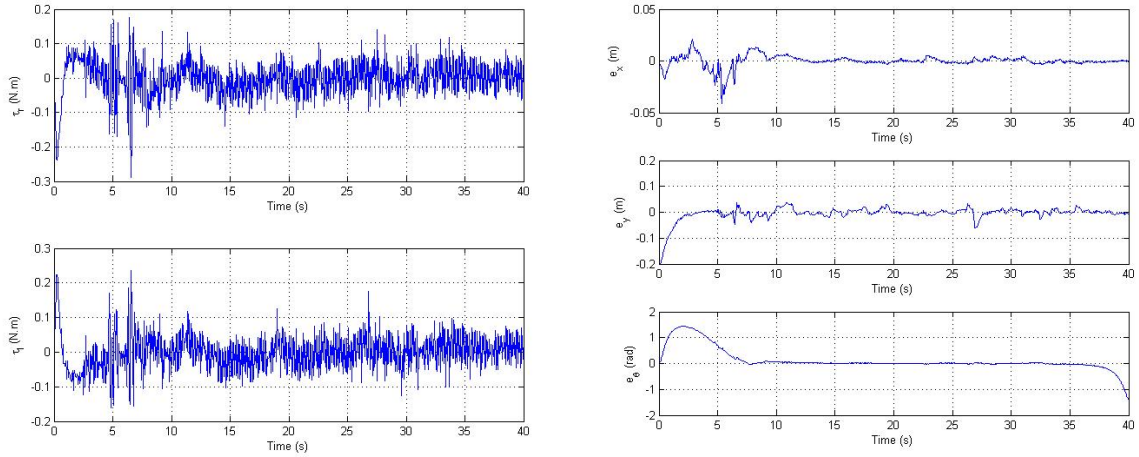
$$\dot{\theta} = \frac{u_2 \cos\theta - u_1 \sin\theta}{L} \tag{3.47}$$

is not controlled. In fact, this control scheme does not use the orientation error. In the case where the application requires to control the orientation, a nonlinear control



(a) Robot's Orientation **(b)** Velocities

Figure 3.19: Robot's orientation and linear and angular velocities



(a) Right and Left Torques **(b)** Tracking errors

Figure 3.20: Applied torques and tracking errors.

is needed. For this purpose, we will investigate the general method of nonlinear control called Immersion and Invariance.

3.4 Motion Control via Immersion and Invariance based approach

The concept of *invariance* has been widely used in control theory. The development of linear and nonlinear geometric control theory has shown that invariant subspaces, and their nonlinear counterpart, invariant distributions, play a fundamental role in the solution of many design problems (see [75], [52]). The notions of invariant distributions and (slow, fast) invariant manifolds have been crucial for the analysis and design problems of linear and nonlinear systems. More precisely, the theory of the center manifold has been instrumental in the design of stabilising control laws for systems with non-controllable linear approximation, see, e.g., [2], whereas the concept of zero dynamics and the strongly related notion of zeroing manifold have been exploited in several local and global stabilisation methods, including passivity-based control [78], backstepping [60] and forwarding [89].

Another, as important, concept has been that of an *immersion*. Its basic idea is to *transform* the system under consideration into a system with pre-specified properties. In more details, a system immersion is a mapping of the initial state from the original state-space to another state-space so as to preserve the input-output map, and is a mapping to a higher dimensional space. For example, the classical problem of immersion of a generic nonlinear system into a linear and controllable system by means of static or dynamic state feedback has been extensively studied, see [52], [75].

The method of Immersion and Invariance (I&I) for stabilization of nonlinear systems was originated in [7], and was further developed in a series of publications that have been recently summarized in [8]. In the I&I approach the desired behavior of the system to be controlled is captured by the choice of a target dynamical system. The control objective is to find a controller which guarantees that the closed-loop system asymptotically behaves like the target system achieving asymptotic model matching. This is formalized by finding a manifold in state-space that can be rendered invariant and attractive, with internal dynamics a copy of the desired closed-loop dynamics, and designing a control law that steers the state of the system toward the manifold.

Our purpose is to investigate a control strategy based on the I&I methodology in order to solve both nonholonomic navigation problems: stabilization about a desired posture and tracking a reference trajectory. We start by introducing the main result of the Immersion and Invariance.

3.4.1 Review of Immersion and Invariance based approach

In this section we briefly present the major result of *Immersion and Invariance* in the following theorem [8].

Theorem 1 *Consider the system*

$$\dot{x} = f(x) + g(x)\,u \tag{3.48}$$

with state $x \in \mathbb{R}^n$ and control $u \in \mathbb{R}^m$ with an equilibrium point $x_* \in \mathbb{R}^n$ to be stabilized. Let $p < n$ and assume we can find mappings

$$\begin{aligned} a &: \mathbb{R}^p \to \mathbb{R}^p \\ \pi &: \mathbb{R}^p \to \mathbb{R}^n \\ c &: \mathbb{R}^n \to \mathbb{R}^m \\ \phi &: \mathbb{R}^n \to \mathbb{R}^{n-p} \\ \psi &: \mathbb{R}^{n\times(n-p)} \to \mathbb{R}^m \end{aligned}$$

such that the following hold

- *(H1)(Target System) The system*

$$\dot{\xi} = \alpha(\xi) \tag{3.49}$$

  with state $\xi \in \mathbb{R}^p$, has an asymptotically stable equilibrium at $\xi_* \in \mathbb{R}^p$ and $x_* = \pi(\xi_*)$

- *(H2)(Immersion condition) For all $\xi \in \mathbb{R}^p$*

$$f(\pi(\xi)) + g(\pi(\xi))\,c(\pi(\xi)) = \frac{\partial \pi}{\partial \xi}\alpha(\xi) \tag{3.50}$$

- *(H3)(Implicit manifold) The set identity*

$$\{x \in \mathbb{R}^n \mid \phi(x) = 0\} = \{x \in \mathbb{R}^n \mid x = \pi(\xi) \ for \ some \ \xi \in \mathbb{R}^p\} \tag{3.51}$$

holds.

- *(H4)(Manifold attractivity and trajectory boundedness) All trajectories of the system*

$$\dot{z} = \frac{\partial \phi}{\partial x}\left(f(x) + g(x)\psi(x,z)\right) \tag{3.52}$$

$$\dot{x} = f(x) + g(x)\psi\left(x,z\right) \tag{3.53}$$

where $z = \phi(x)$, are bounded and satisfy

$$\lim_{t\to\infty} z(t) = 0 \tag{3.54}$$

Then, x_* is an asymptotically stable equilibrium of the closed loop system

$$\dot{x} = f\left(x\right) + g\left(x\right)\psi\left(x,\phi(x)\right) \tag{3.55}$$

In fact, if conditions *H1* to *H4* hold, then any trajectory $x(t)$ of closed-loop system (3.55) is the image through the mapping $\pi(\Delta)$ of a trajectory $\xi(t)$ of target system (3.49), as illustrated in Fig. 3.21.

In standard applications of I&I, the target system is a priori defined, hence condition (*H1*) is automatically satisfied. Given the target system, equation (3.50) of condition (*H2*) defines a set of partial differential equations (PDEs) in the unknown function $\pi(.)$, where $c(.)$ is a free parameter. However, in [1] a procedure was proposed to obviate the solution of the PDEs for a class of underactuated mechanical systems and was demonstrated for the cart–pendulum system. More specifically, it was proposed to leave α as a free parameter and view PDEs (3.50) as algebric equations relating α with π (and its partial derivatives). Then suitable expressions for α were selected so that the desired stability properties for the target dynamics were ensured.

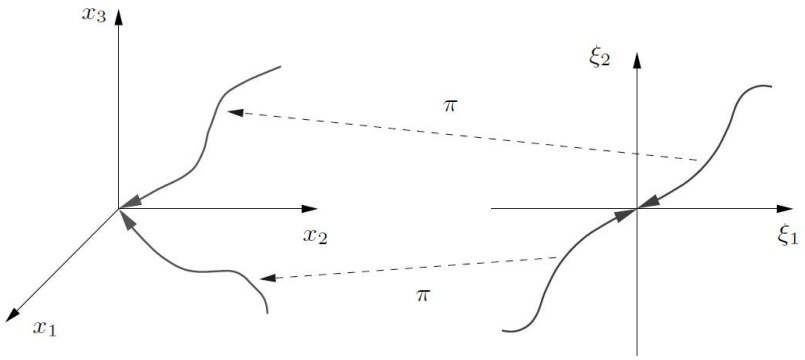


Figure 3.21: Graphical illustration of the mapping between the trajectories of the system to be controlled and the target system for $p = 2$ and $n = 3$.

3.4.2 Posture Stabilization

Let's consider the kinematic model in chained forms

$$\Sigma : \begin{cases} \dot{x}_1 &= u_1 \\ \dot{x}_2 &= u_2 \\ \dot{x}_3 &= x_2 u_1 \end{cases} \tag{3.56}$$

The key idea is to immerse a two dimensional system which describes a desired dynamics, into three dimensional one (3.56). Thus, we define the target dynamics as:

$$\Sigma_T : \begin{cases} \dot{\xi}_1 &= -\alpha_1 \xi_1 \\ \dot{\xi}_2 &= -\alpha_2 \xi_2 \end{cases} \tag{3.57}$$

where $\alpha_1, \alpha_2 > 0$.

A possible choice of the mapping π is:

$$\pi(\xi) = \begin{bmatrix} \xi_1 \\ \xi_2 \\ \pi_3(\xi_1, \xi_2) \end{bmatrix} \tag{3.58}$$

By applying the immersion condition (*H2*) we obtain:

$$c_1(\pi(\xi)) = -\alpha_1 \xi_1 \tag{3.59}$$

$$c_2(\pi(\xi)) = -\alpha_2 \xi_2 \tag{3.60}$$

$$\xi_2 \cdot c_1(\pi(\xi)) = -\alpha_1 \frac{\partial \pi_3}{\partial \xi_1} \cdot \xi_1 - \alpha_2 \frac{\partial \pi_3}{\partial \xi_2} \cdot \xi_2 \tag{3.61}$$

from (3.59) and (3.61) we get:

$$-\alpha_1\xi_1\xi_2 = -\alpha_1\frac{\partial\pi_3}{\partial\xi_1}\cdot\xi_1 - \alpha_2\frac{\partial\pi_3}{\partial\xi_2}\cdot\xi_2 \tag{3.62}$$

The solution of equation (3.62) is

$$\pi_3(\xi_1,\xi_2) = \frac{b}{b+1}\xi_1\xi_2 + \beta_1(\xi_1\xi_2^{-b}) \tag{3.63}$$

where $b = \alpha_1/\alpha_2$, and $\beta_1(.)$ is an arbitrary differentiable function.

Now, we need to verify the remaining conditions of Theorem 1. It is straightforward to define the manifold of (*H3*) via the function:

$$\phi(x) \triangleq x_3 - \pi_3(x_1,x_2) \tag{3.64}$$

In order to satisfy (*H4*) and render the manifold attractive, we examine the distance of the system trajectories to the manifold, defined as $z \triangleq \phi(x)$ and called the off-the-manifold coordinate, with its dynamics given as

$$\begin{aligned}\dot{z} =& \dot{x}_3 - \dot{\pi}_3(x_1,x_2)\\ =& x_2\cdot\psi_1(x,z) - \frac{\partial\pi_3}{\partial x_1}\cdot\psi_1(x,z) - \frac{\partial\pi_3}{\partial x_2}\cdot\psi_2(x,z)\\ =& \left(\frac{1}{b+1}x_2 - x_2^{-b}\beta_1'\left(x_1x_2^{-b}\right)\right)\cdot\psi_1(x,z)\\ &- \left(\frac{b}{b+1}x_1 - bx_1x_2^{-b-1}\beta_1'\left(x_1x_2^{-b}\right)\right)\psi_2(x,z)\end{aligned} \tag{3.65}$$

We observe that the control

$$\begin{cases}\psi_1 = \dfrac{-\left(\frac{\alpha_1}{b+1}x_1x_2 - \alpha_1x_1x_2^{-b}\beta_1'\left(x_1x_2^{-b}\right)\right) - \gamma(x_3-\pi_3)}{\left(\frac{1}{b+1}x_2 - x_2^{-b}\beta_1'\left(x_1x_2^{-b}\right)\right)}\\ \psi_2 = -\alpha_2x_2\end{cases} \tag{3.66}$$

fixes the off-the-manifold dynamics to $\dot{z} = -\gamma z$, hence selecting $\gamma > 0$ drives z to zero exponentially with a rate of convergence γ.

One interesting choice of β_1 is $\beta_1 \equiv 0$, in this case we have $x_* = \pi(\xi_*) = (0,0,0)$, and the corresponding controller takes the form

$$\begin{cases}u_1 = \lambda_1x_1 + \lambda_2\dfrac{x_3}{x_2}\\ u_2 = -\alpha_2x_2\end{cases} \tag{3.67}$$

where $\lambda_1 = b(\gamma - \alpha_2)$ and $\lambda_2 = -\gamma(b+1)$.

Remark 4 *Discontinuous control law (3.67) obtained using the method of Immersion and Invariance is similar to the controller described in ([4] and [5]) for nonholonomic systems in chained forms, the difference is the choice of $u_2 = -\alpha_2 x_2$ instead of $u_1 = -\alpha_1 x_1$ as in Astolfi's controller. This choice has been made for practical reasons.*

The closed-loop system is then

$$\Sigma : \begin{cases} \dot{x}_1 &= \lambda_1 x_1 + \lambda_2 \frac{x_3}{x_2} \\ \dot{x}_2 &= -\alpha_2 x_2 \\ \dot{x}_3 &= \lambda_1 x_1 x_2 + \lambda_2 x_3 \end{cases} \tag{3.68}$$

Lemma 1 *The system of ordinary differential equations (3.68), with initial condition $[x_1(0)\ x_2(0)\ x_3(0)]^T$ such that*

$$x_2(0) \neq 0 \tag{3.69}$$

has a unique and well defined solution for all $t \geq 0$.

Moreover, let

$$\Lambda = \begin{bmatrix} \lambda_1 & \lambda_2 \\ \lambda_1 & \lambda_2 + \alpha_2 \end{bmatrix} \tag{3.70}$$

and

$$\Psi(t) = e^{\Lambda t} \tag{3.71}$$

Then the unique solution of the system of o.d.e.(3.68) with initial condition $\boldsymbol{x}(0)$ satisfying condition (3.69) is

$$x_2(t) = x_2(0)e^{(-\alpha_2 t)} \tag{3.72}$$

$$\begin{bmatrix} x_1(t) \\ x_3(t) \end{bmatrix} = \Gamma(t)\Psi(t)\tilde{x}(0) \tag{3.73}$$

where

$$\begin{aligned} \tilde{x}(0) &= \begin{bmatrix} x_1(0) \\ \frac{x_3(0)}{x_2(0)} \end{bmatrix} \\ \Gamma(t) &= \begin{bmatrix} 1 & 0 \\ 0 & x_2(t) \end{bmatrix} \end{aligned} \tag{3.74}$$

Proof 1 *(see [4])*

For the component x_2, the proof is trivial. Regarding the remaining components of the state vector, we note that if $x_2 \neq 0$ it is possible to apply the state transformation

$$\begin{bmatrix} p_1 \\ p_3 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_3/x_2 \end{bmatrix} \tag{3.75}$$

yielding

$$\begin{bmatrix} \dot{p}_1 \\ \dot{p}_3 \end{bmatrix} = \Lambda \begin{bmatrix} p_1 \\ p_3 \end{bmatrix} \tag{3.76}$$

System (3.76), with initial conditions (3.74), admits the closed integral

$$\begin{bmatrix} x_1(t) \\ x_3(t) \end{bmatrix} = \Psi(t)\tilde{x}(0) \tag{3.77}$$

Hence, the claim directly follows, applying the inverse state transformation □.

The main result of Lemma 1 is that discontinous control law (3.67) is well defined and bounded, for all $t \geq 0$, along the trajectories of closed-loop system (3.68) with initial condition $[x_1(0)\ x_2(0)\ x_3(0)]^T$ such that $x_2(0) \neq 0$, if and only if the matrix Λ has all eigenvalues with negative real part. which corresponds to choosing $\alpha_2 < \gamma$.

Remark 5 *The choice of function β_1 enables us to derive a class of controllers for system (3.56). However, the target equilibrium manifold will depend on the function β_1, and/or the initial conditions of x_1 and x_2.*

3.4.3 Trajectory Tracking

As we have seen in chapter 1, in order for a trajectory to be feasible, it must satisfy the nonholonomic constraint on the vehicle motion or, in other words, it must satisfy the equations

$$\begin{array}{rcl} \dot{x}_d & = & v_d \cos\theta_d \\ \dot{y}_d & = & v_d \sin\theta_d \\ \dot{\theta}_d & = & \omega_d \end{array} \tag{3.78}$$

for some choice of reference inputs v_d and ω_d. In the following we will assume that

- (a1) v_d and ω_d are continuous and bounded;
- (a2) there exists ϵ such that $v_d(t) \geq \epsilon > 0 \ \ \forall t \geq 0$.

By comparing the desired state $\mathbf{q}_d = [x_d(t)\ y_d(t)\ \theta_d(t)]^T$ with the current measured state $\mathbf{q} = [x(t)\ y(t)\ \theta(t)]^T$, it is possible to compute an error vector that can be fed to

the controller. However, rather than using directly the difference between $\mathbf{q}_d$ and $\mathbf{q}$, it is convenient to define the tracking error as

$$\mathbf{e} = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_d - x \\ y_d - y \\ \theta_d - \theta \end{bmatrix} \tag{3.79}$$

And the error dynamics are [58]

$$\begin{bmatrix} \dot{e}_1 \\ \dot{e}_2 \\ \dot{e}_3 \end{bmatrix} = \begin{bmatrix} \omega_d e_2 \\ v_d \sin e_3 - \omega_d e_1 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & -e_2 \\ 0 & e_1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \eta_1 \\ \eta_2 \end{bmatrix} \tag{3.80}$$

with

$$\begin{aligned} \eta_1 &= v_d \cos e_3 - v \\ \eta_2 &= \omega_d - \omega \end{aligned}$$

In what follows, we will only consider the local tracking problem, in this case we have small initial tracking errors $(e_1(0), e_2(0), e_3(0))$. Let's define the following target dynamics

$$\Sigma_T : \ \{ \ \dot{\xi} \ = \ \alpha(\xi) \tag{3.81}$$

where $\alpha(.)$ is a scalar function to be defined.

A possible choice of the mapping π is the form:

$$\pi(\xi) = \begin{bmatrix} 0 \\ \xi \\ \pi_3(\xi) \end{bmatrix} \tag{3.82}$$

By applying immersion condition (*H2*), we thus obtain

$$\omega_d \xi + c_1(\pi(\xi)) - \xi c_2(\pi(\xi)) = 0 \tag{3.83}$$

$$v_d \sin \pi_3(\xi) = \alpha(\xi) \tag{3.84}$$

$$c_2(\pi(\xi)) = \frac{\partial \pi_3}{\partial \xi} \alpha(\xi) \tag{3.85}$$

Choose then $\pi_3(\xi) = -\beta_2(\xi)$ such that the following conditions hold

- (c1) $\beta_2 : \mathbb{R} \to]-\pi, \pi[$, $\beta_2 \in C^\infty$.
- (c2) $\beta_2(0) = 0$, $x\beta_2(x) > 0 \ \ \forall x \neq 0$.
- (c3) β_2' is bounded.

Using equation (3.84) yields the corresponding target dynamics

$$\dot{\xi} = -v_d \sin \beta_2(\xi) \tag{3.86}$$

We note that, under conditions (c1) and (c2) and assumptions (a1) and (a2), $-v_d \xi \sin \beta_2(\xi) < 0 \quad \forall \xi \neq 0$, which means that $\xi_* = 0$ is a stable equilibrium of system (3.86), and we have $\mathbf{e}_* = \pi(\xi_*) = \mathbf{0}$.

An implicit definition of the manifold $\phi(\mathbf{e}) = 0$ is obtained by selecting

$$\phi(e) = \begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix} \triangleq \begin{bmatrix} e_1 \\ e_3 - \pi_3(e_2) \end{bmatrix} \tag{3.87}$$

In order to satisfy condition (*H4*), let's consider the candidate Lyapunov function

$$V(e_1, e_2, z_2) = \frac{1}{2}(z_1^2 + e_2^2) + \frac{1}{2\gamma} z_2^2 \tag{3.88}$$

with $z_1 = e_1$, $z_2 = e_3 - \pi_3(\xi)|_{\xi = e_2}$ and $\gamma > 0$. As can be directly verified, V is positive definitive and radially unbounded. Taking the derivative of V along trajectories of system (3.80)

$$\begin{aligned} \dot{V}(e_1, e_2, z_2) =& e_1(\omega_d e_2 + \psi_1(\mathbf{e}, z) - e_2 \psi_2(\mathbf{e}, z)) \\ &+ e_2(v_d \sin(z_2 - \beta_2(e_2)) - w_d e_1 + e_1 \psi_2(\mathbf{e}, z)) \\ &+ \frac{1}{\gamma} z(\psi_2(\mathbf{e}, z) + \beta_2'(v_d \sin(e_3) - w_d e_1 + e_1 \psi_2(\mathbf{e}, z))) \end{aligned} \tag{3.89}$$

Noting that

$$\sin(z_2 - \beta_2(e_2)) = -\sin(\beta_2(e_2)) + z_2 \mu(e_2, z_2) \tag{3.90}$$

where

$$\mu(e_2, z_2) = \cos(\beta_2(e_2)) \frac{\sin z_2}{z_2} + \sin(\beta_2(e_2)) \cdot \frac{1 - \cos z_2}{z_2} \tag{3.91}$$

is defined for $z_2 \in \mathbb{R}$. It follows that

$$\begin{aligned} \dot{V}(e_1, e_2, z_2) =& e_1 \psi_1 - v_d e_2 \sin(\beta_2(e_2)) \\ &+ \frac{1}{\gamma} z_2[(1 + \beta_2'(e_2) e_1) \psi_2 + \gamma v_d \mu e_2 + \beta_2'(e_2)(v_d \sin(e_3) - w_d e_1)] \end{aligned} \tag{3.92}$$

By choosing

$$\begin{aligned} \psi_1(\mathbf{e}, z_2) =& -k_1 e_1 \\ \psi_2(\mathbf{e}, z_2) =& \frac{1}{1 + \beta_2'(e_2) e_1}[-\gamma v_d \mu e_2 + \beta_2'(e_2)(-v_d \sin(e_3) + w_d e_1) - k_2 \gamma z_2] \end{aligned} \tag{3.93}$$

with $k_1, k_2 > 0$, we have

$$\begin{aligned}\dot{V}(\mathbf{e}, z_2) &= -k_1 e_1^2 - v_d e_2 \sin(\beta_2(e_2)) - k_2 z_2^2 \\ &\leq -k_1 e_1^2 - \epsilon e_2 \sin(\beta_2(e_2)) - k_2 z_2^2 \leq 0\end{aligned} \tag{3.94}$$

Let's define the set

$$\Omega(\beta_2') = \left\{(e_1, e_2, z_2) \in \mathbb{R}^3 : \left\|\beta_2'\right\|_\infty \cdot |e_1| < 1\right\}$$

Also, let M be a set given by

$$M = \left\{(e_1, e_2, z_2) \in \mathbb{R}^3 : V(\mathbf{e}, z) < c^*, \forall t \geq 0\right\}$$

where c^* is the largest constant c such that

$$\left\{(e_1, e_2, z_2) \in \mathbb{R}^3 : V(\mathbf{e}, z) < c\right\} \subset \Omega(\beta_2{}')$$

For any initial condition $(e_1(0), e_2(0), z_2(0)) \in M$, it follows from (3.94) that ($e_1(t)$, $e_2(t)$, $z_2(t)$) remains in M, and therefore ψ_2 is well defined on $t \in [0, \infty[$. It follows also from (3.94) that $z_1 = e_1$, e_2 and z_2 are bounded and converge to zero, and since $e_3 = z_2 - \beta_2(e_2)$, we conclude that e_3 is bounded. Then it follows from Theorem 1 that $\mathbf{e}_* = \mathbf{0}$ is an asymptotically stable equilibrium of the closed loop system $\square$.

A possible choice of $\beta_2(\xi)$ is $\beta_2(\xi) = \text{atan}(r\xi)$ with $r > 0$, we note that it satisfies conditions (c1) to (c3). In this case, using (3.93) yields the corresponding control law

$$\begin{aligned}\eta_1(\mathbf{e}, z) &= -k_1 e_1 \\ \eta_2(\mathbf{e}, z) &= \frac{1 + r^2 e_2^2}{1 + r^2 e_2^2 + r e_1}(-\gamma v_d \mu e_2 + \frac{r}{1 + r^2 e_2^2}(-v_d \sin(e_3) + w_d e_1) - k_2 \gamma z_2)\end{aligned} \tag{3.95}$$

with $z_2 = e_3 + \text{atan}(r e_2)$.

3.4.4 Simulation Results

In order to verify the performance of the proposed control laws we first carried out numerical simulations for both control problems

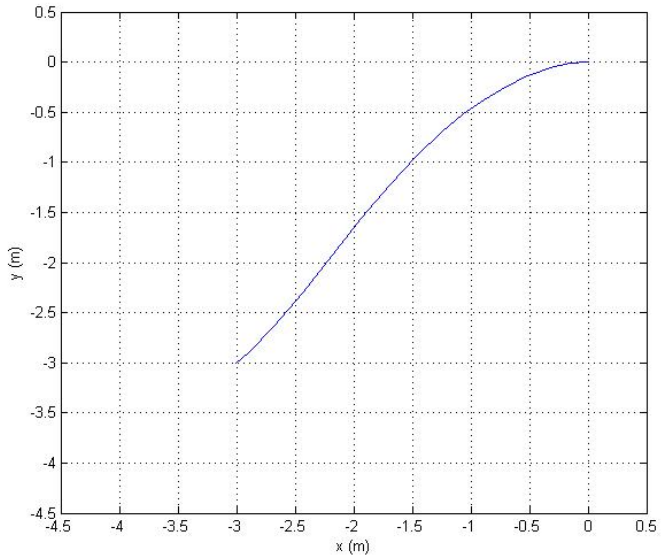


Figure 3.22: Stabilization of robot system using $I\&I$ controller for $(x_0, y_0, \theta_0) = (-1, -1, \pi/4)$.

A. Posture Stabilization

In this case, the desired posture is the origin $(x_d, y_d, \theta_d) = (0, 0, 0)$. At $t = 0$ sec, the initial position of the robot is $(x(0), y(0), \theta(0)) = (-3, -3, \pi/4)$. We applied controller (3.67) and control gains are tuned in order to achieve fast convergence towards the desired manifold and to satisfy limitations on linear and angular velocities, and in our simulations they are set to $k = 1$,$\lambda_1 = 1.4$ and $\lambda_2 = -5$. As we can observe from Figs. 3.22 through 3.25 the controller ensures the stabilization of the mobile robot.

B. Trajectory Tracking

The tracking performance of this control strategy is verified in the tracking of the reference trajectory that consists of successive straight line segments and arc line segments which may represent the ouput of a typical path planner [61]. Controller parameters are set to $k_1 = 0.8$, $k_2 = 0.8$ and $\gamma = 1$. Reference and actual trajectories of the robot are depicted in Fig. 3.26. Tracking on X-Axis, Y-axis and robot's orientation are shown in Fig. 3.27, while tracking errors are shown in Fig. 3.28. Clearly the mobile robot

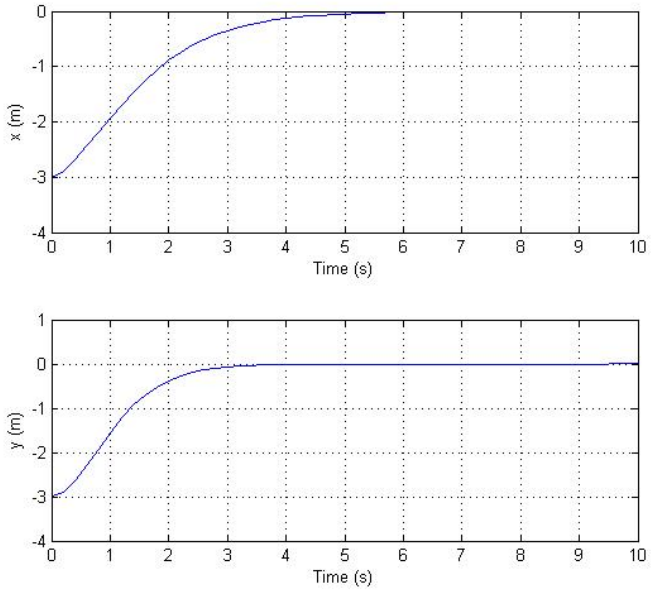


Figure 3.23: Stabilization of robot system using $I\&I$ controller for $(x_0, y_0, \theta_0) = (-1, -1, \pi/4)$: Time responses.

is able to track almost perfectly the reference trajectory and the inputs remain within acceptable limits (Fig. 3.29).

3.5 Conclusion

In this chapter, we have discussed the motion control problem for a nonholonomic mobile robot, with reference to two basic motion tasks, i.e., posture regulation and trajectory tracking. Two control approaches have been presented: First, state feedback Linearization is extended to include both kinematic and dynamic models, or what is called second-order kinematic model. A controller is derived, on the basis of the Integral Sliding Mode approach with the objective of posture regulation and tracking pre-generated trajectories. This combination has led to satisfactory results in terms of stabilization and robustness in presence of parameters uncertainties, unmodeled dynamics and unknown bounded disturbances. Second, a control strategy based on the application of *Immersion and Invariance* methodology is described. The proposed control strategy guarantees that the closed-loop system asymptotically behaves like a

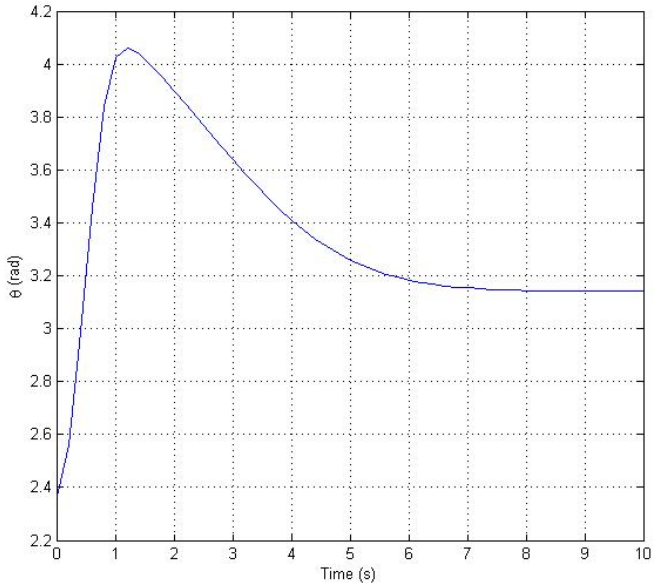


Figure 3.24: Stabilization of robot system using $I\&I$ controller for $(x_0, y_0, \theta_0) = (-1, -1, \pi/4)$: Robot's Orientation.

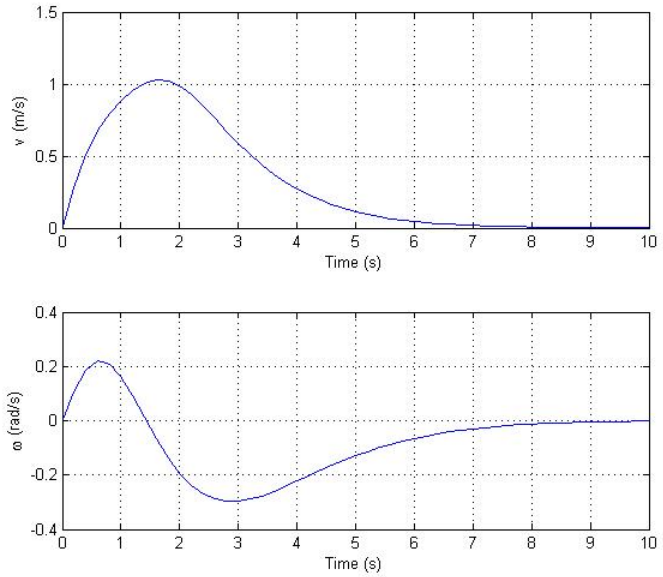


Figure 3.25: Stabilization of robot system using $I\&I$ controller for $(x_0, y_0, \theta_0) = (-1, -1, \pi/4)$: Linear and angular velocities.

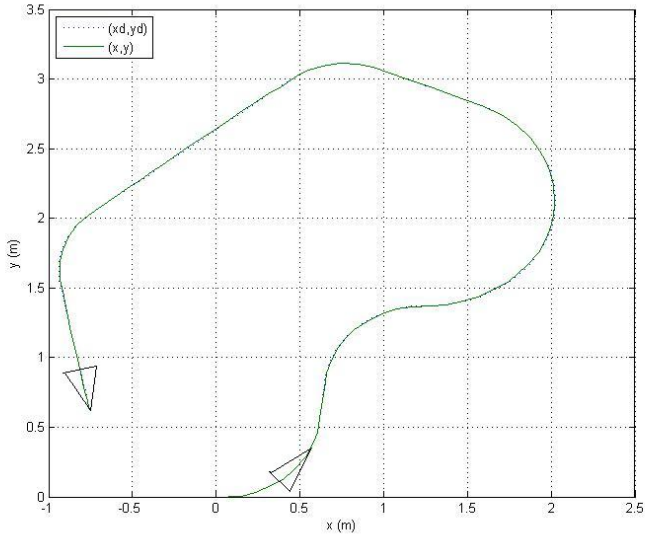


Figure 3.26: Trajectory tracking of robot system using I&I controller.

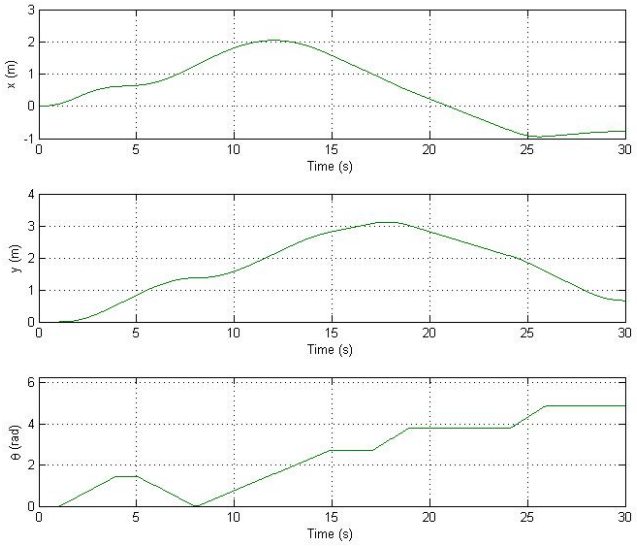


Figure 3.27: Trajectory tracking of robot system using I&I controller: Tracking on x, y and θ

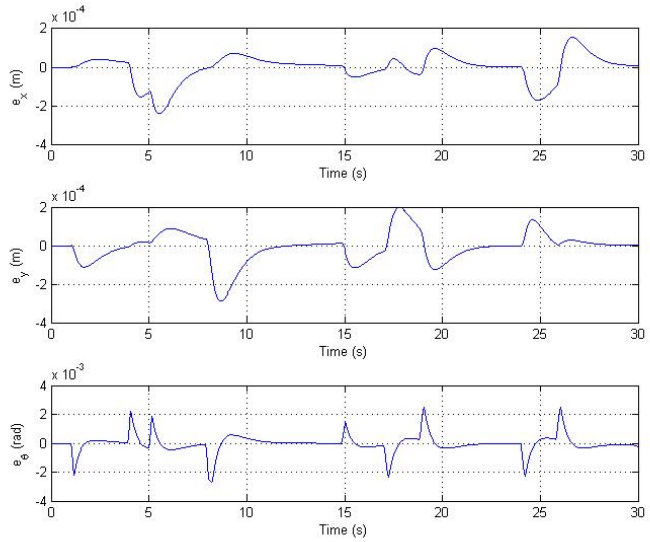


Figure 3.28: Trajectory tracking of robot system using I&I controller: Tracking errors

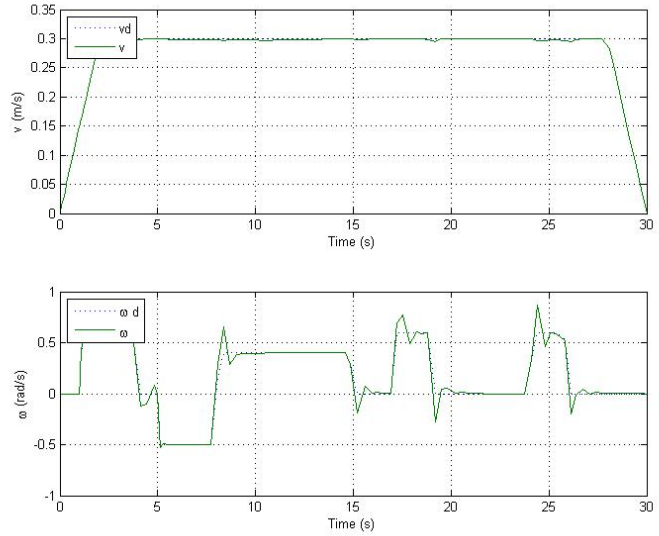


Figure 3.29: Trajectory tracking of robot system using I&I controller: Velocities

given target system achieving asymptotic model matching, which makes the system performance adjustment simpler and physically meaningful. Stability of the system has been guaranteed by appropriate choice of the target systems. Simulations have been conducted which established the quality of both control strategies.

Chapter 4

Visual Servoing

Visual servoing is a technique which consists of controlling the movements of a robotic system using visual information, from one or more cameras (or more generally a vision sensor) embedded within the system, or outside the system in some cases.

A key characteristic of visual servoing, compared to motion and force control, is that the controlled variables are not directly measured by the sensor, but are obtained from the measured quantities through complex elaborations, based on algorithms of *image processing* and *computational vision*. A camera provides a two-dimensional matrix of values of light intensity. From this matrix, the so-called image feature parameters are to be extracted in real time. The geometric relationships between one or more two-dimensional views of a scene and the corresponding 3D space are the basis of techniques of *pose estimation* of the robot with respect to the surrounding objects. In this regard, of fundamental importance is the operation of camera calibration, which is necessary for calculating the intrinsic parameters, relating the quantities measured in the image plane to those referred to the camera frame, and the extrinsic parameters, relating the latter to quantities defined in a frame attached to the robot.

The vision-based control schemes can be divided into two categories, namely, those that realize visual servoing in operational space, also termed *position-based visual servoing* (3D), and those that realize visual servoing in the image space, also known as *image-based visual servoing* (2D). The main difference lies in the fact that the schemes of the first category use visual measurements to reconstruct the relative pose of the object with respect to the robot, or vice versa, while the schemes of the second category are based on the comparison of the feature parameters of the image of the object

between the current and the desired pose. There are also schemes combining characteristics common to both categories, that can be classified as *hybrid visual servoing*.

In visual servoing literature, visual system has been classified depending on the number of cameras involved to: *mono-vision* (one camera) and *stereo vision* (two or more cameras). Mono-vision system is classified depending on the camera's location to: *eye-to-hand*, where the camera is mounted in a fixed location, and the mobile configuration, and *eye-in-hand*, with the camera attached to the robot. Our approach to visual servoing is based upon a single eye-in-hand camera.

All visual servoing schemes depend on the actual knowledge of image feature parameters, which makes the extraction of image features a key step to visual servoing.

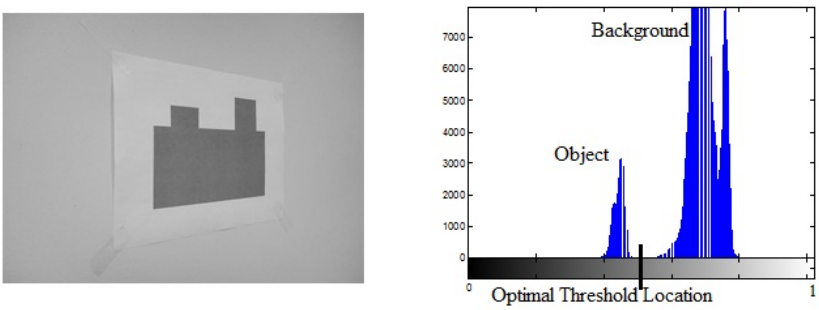
4.1 Extraction of Visual Features

The task of a camera as a vision sensor is to measure the intensity of the light emitted or reflected by an object. To this end, a photosensitive element, termed pixel (or photosite), is employed, which is capable of transforming light energy into electric energy. Different types of sensors are available depending on the physical principle exploited to realize the energy transformation. The most widely used devices are CCD and CMOS sensors based on the photoelectric effect of semiconductors.

An image contains a large amount of data represented as pixels. Much of this data does not contain relevant information. As part of visual servoing, it is essential to identify the portions of the image that carry information relevant to the type of image features that we seek to identify in the scene. It is also necessary to represent this information concisely in system memory. To this end, two basic operations are required: image segmentation and image interpretation [92].

4.1.1 Image Segmentation

Segmentation consists of a grouping process, by which the image is divided into a certain number of groups, referred to as segments, so that the components of each group are similar with respect to one or more characteristics. Typically, distinct segments of the image correspond to distinct objects of the environment, or homogeneous object parts. There are two approaches to the problem of image segmentation:



(a) **(b)**

Figure 4.1: Gray-level image and corresponding gray-level histogram on the right

1. Region-based segmentation whose objective is that of grouping sets of pixels sharing common features into two-dimensional connected areas, with the implicit assumption that the resulting regions correspond to real-world surfaces or objects. In many applications of practical interest a thresholding approach is adopted and a light intensity scale composed of only two values (0 and 1) is considered. This operation is referred to as binary segmentation or image binarization, and corresponds to separating one or more objects present in the image from the background by comparing the gray level of each pixel with a threshold l. A crucial factor for the success of binary segmentation is the choice of the threshold. A widely adopted method for selecting the threshold is based on the gray-level histogram, under the assumption that it contains clearly distinguishable minimum and maximum values, corresponding to the gray levels of the objects and of the background (see Fig. 4.1). This method of segmentation is fast but requires a large memory storage.

2. Boundary-based segmentation is aimed at identifying the pixels corresponding to object contours and isolating them from the rest of the image. The boundary of an object, once extracted, can be used to define the position and shape of the object itself. In the case of simple and well-defined shapes, boundary detection becomes straightforward and segmentation reduces to the sole edge detection. Several edge detection techniques exist. Most of them require the calculation of the gradient or of the Laplacian of the image [42]. This method requires less memory storage

since boundaries contain a reduced number of pixels, but is relatively slow since it requires more complex computing.

4.1.2 Image Interpretation

Image interpretation is the process of calculating the image feature parameters from the segments, whether they are represented in terms of boundaries or in terms of regions.

The feature parameters used in visual servoing applications sometimes require the computation of the so-called *moments*. Image moments and various types of moment-based invariants play very important role in object recognition and shape analysis [50], [41], [70]. These parameters are defined on a region $\mathcal{R}$ of the image and can be used to characterize the position, orientation and shape of the two-dimensional object corresponding to the region itself. The (i,j)th order geometric moment of a region $\mathcal{R}$ on a grey-level image I(X,Y) is defined as

$$m_{i,j} = \iint_{\mathcal{R}} I(X,Y) X^i Y^j dX dY \tag{4.1}$$

In the case of a digital image, equation (4.1) becomes

$$m_{i,j} = \sum_{X,Y \in \mathcal{R}} I(X,Y) X^i Y^j \tag{4.2}$$

where $I(X,Y)$ is the gray level of an individual pixel.

In the case of binary images, by assuming the light intensity equal to one for all the points of region $\mathcal{R}$, and equal to zero for all the points not belonging to $\mathcal{R}$, the following simplified definition of moment is obtained

$$m_{i,j} = \sum_{X,Y \in \mathcal{R}} X^i Y^j \tag{4.3}$$

From this definition we see that moment $m_{0,0}$ coincides with the area of the region, computed in terms of the total number of pixels of region $\mathcal{R}$.

The quantities

$$X_g = \frac{m_{1,0}}{m_{0,0}}, \qquad Y_g = \frac{m_{0,1}}{m_{0,0}} \tag{4.4}$$

define the coordinates of the so-called *centroid* of the region. These coordinates can be used to detect uniquely the position of region $\mathcal{R}$ on the image plane. Many kinds of

image-based invariants have been studied in the literature and can be used to deduce more parameters of the region such as shape and orientation (see Appendix B.2).

The advantage of using image moments in visual servoing comes from the fact that they provide a generic representation of any object, with simple or complex shapes, that can be segmented in an image. They also provide a more geometric and intuitive meaning than other features [23]. Moreover, the fact that image moments are computed on the whole region makes them less vulnerable to image noise and other measurement errors. This, however, comes with a heavy cost of computational complexity, since direct calculation of discrete moments by (4.3) is time consuming.

Fortunately, this problem has been the subject of many studies in the field of computer vision and image processing, a large amount of effort has been spent in the past to develop more effective algorithms, e.g., see [107], [24], [63]. Particular attention has been paid to binary images because of their importance in practical pattern recognitions. Since any binary object is fully determined by its boundary, various boundary-based methods of moments calculation have been developed. Among these are a group of methods that are based on Green theorem, which evaluates the double integral over the object by means of single integration along the object boundary [64],[53],[81]. There are also methods that are based on polygonal approximation of the object boundary. Object moments are then calculated via the corner points [62], [93].

4.2 Camera Modelling and Calibration

We mean by *camera model*, the set of geometric laws defining how, during the process of capturing an image, a point in the three dimensional space is projected on a two-dimensional plane. A model is characterized by a number of parameters that allow us to calculate the coordinates in pixels of a point projection on the image, using its Cartesian coordinates in 3D space. Depending on the desired accuracy, several models have been proposed taking into account more or less faithfully, the way the light beam moves to form the image [37], [104], [46].

4.2.1 Pinhole Camera and Perspective Projection

The pinhole camera model shown in Fig. 4.2 consists of a plane $\mathcal{F}$ at a fixed distance f in front of an *image plane* $\mathcal{I}$. An ideal pinhole C is found in the plane $\mathcal{F}$. The plane $\mathcal{F}$ is

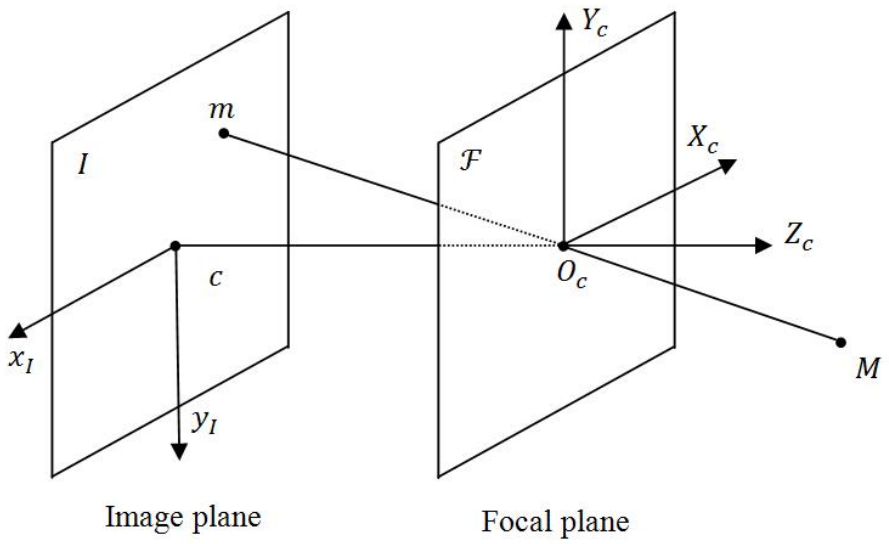


Figure 4.2: The pinhole camera model.

called the *focal plane*, and the image plane is also called the *retinal plane*. We assume that an enclosure is provided so that only light coming through the pinhole can reach the image plane. The rays of light reflected (or emitted) by an object pass through the pinhole and form an inverted image of that object on the image plane. Each point in the object, its corresponding image point and the pinhole constitute a straight line. This kind of projection from 3D space to plane is called *perspective projection*.

The point C is called the *optical center*, and the distance between the optical center and the image plane f is called the *focal length*. The line going through the optical center C and perpendicular to the image plane $\mathcal{I}$ is called the *optical axis*. Experiences have shown that such a simple system can accurately model the geometry and optics of most of the modern video cameras [104].

For computational ease, let's consider an inverted virtual image plane positioned before the lens, in correspondence of the plane $z_c = f$ of the camera frame. The projection of a point in space on an image depends, on one hand, on its location relative to the camera and on the other hand on the physical characteristics of the camera. Let's start by defining the following coordinate systems (Fig. 4.3).

- Base frame or world frame $O_b\{X, Y, Z\}$ where the 3D points of the object being

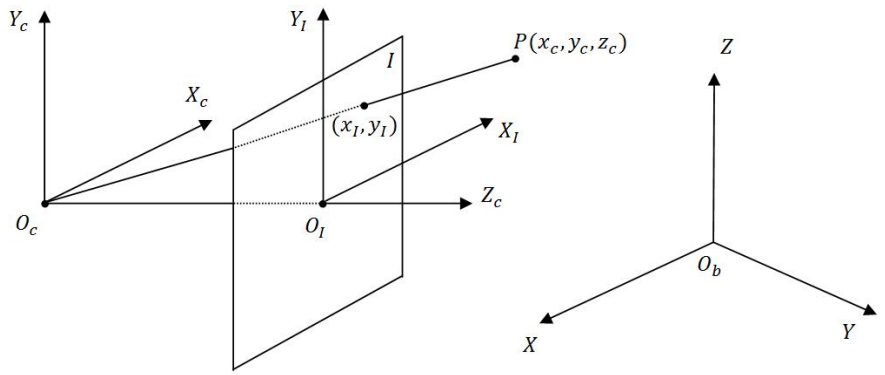


Figure 4.3: Frontal perspective transformation.

filmed are expressed.

- Camera frame $O_c\{X_c, Y_c, Z_c\}$, where the origin is at the optical center and the Z_c-axis coincides the optical axis of the camera.
- Image frame $O_I\{X_I, Y_I\}$ for the image plane is defined such that the origin is at the point c (intersection of the optical axis with image plane) and that the axes are determined by the camera scanning and sampling system.

Let $\mathbf{p}_b = [x \;\; y \;\; z]_b^T$ be an arbitrary point of the object expressed in the base frame. It is expressed in the camera frame as $\mathbf{p}_c[x_c \;\; y_c \;\; z_c]^T$. By adopting the homogeneous representation of coordinates we have the relationship

$$\tilde{\mathbf{p}}_c = \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} = \mathbf{T}_b^c \tilde{\mathbf{p}}_b = \mathbf{T}_b^c \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{4.5}$$

where $\mathbf{T}_b^c \in \mathbb{R}^{4\times4}$ is the homogeneous transformation matrix from base frame to camera frame. $\mathbf{T}_b^c$ is composed of rotation matrix $\mathbf{R}_b^c \in \mathbb{R}^{3\times3}$ and a translation vector $\mathbf{t}_b^c \in \mathbb{R}^{3\times1}$.

$$\mathbf{T}_b^c = \begin{bmatrix} \mathbf{R}_b^c & \mathbf{t}_b^c \\ 0 & 1 \end{bmatrix} \tag{4.6}$$

$\mathbf{R}_b^c$ and $\mathbf{t}_b^c$ are called the *extrinsic parameters*, and $\mathbf{T}_b^c$ is the matrix of extrinsic parameters.

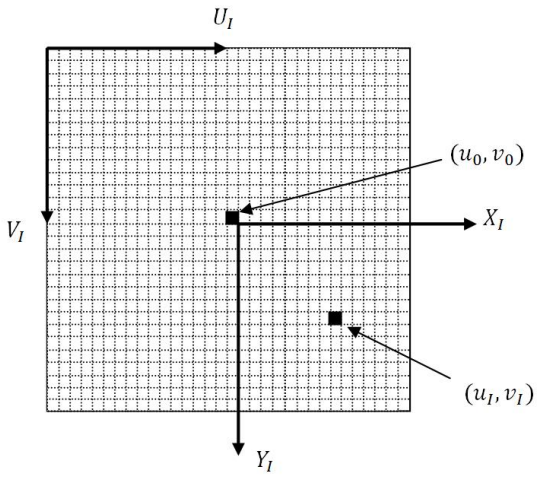


Figure 4.4: Passing from metric image coordinates to pixels image coordinates.

Due to the refraction phenomenon, the point in the camera frame is transformed into a point in the virtual image plane via the *perspective transformation*, i.e.,

$$\begin{aligned} x_I &= \frac{f x_c}{z_c} \\ y_I &= \frac{f y_c}{z_c} \end{aligned} \tag{4.7}$$

A visual information is typically elaborated by a digital processor, and thus the measurement principle is to transform the light intensity $\mathbf{p}_I(x_I, y_I)$ of each point in the image plane into a number. It is clear that a spatial sampling is needed since an infinite number of points in the image plane exist, as well as a temporal sampling since the image can change during time. The CCD or CMOS sensors play the role of spatial samplers, while the shutter in front of the lens plays the role of the temporal sampler.

The spatial sampling unit is the pixel, and thus the coordinates $\mathbf{p}_I(x_I, y_I)$ of a point in the image plane are to be expressed in pixels, i.e., $\mathbf{s}_I(u_I, v_I)$. Due to the photosite finite dimensions, the pixel coordinates of the point are related to the coordinates in metric units through two scale factors K_u and K_v

$$\begin{aligned} u_I &= K_u x_I + u_0 \\ v_I &= K_v y_I + v_0 \end{aligned} \tag{4.8}$$

where u_0 and v_0 are the offsets which take into account the position of the origin of the pixel coordinate system with respect to the optical axis (see Fig. 4.4).

From (4.7) and (4.8) we get

$$\begin{aligned} u_I &= \frac{\alpha_u x_c}{z_c} + u_0 \\ v_I &= \frac{\alpha_v x_c}{z_c} + v_0 \end{aligned} \tag{4.9}$$

where $\alpha_u = fK_u$ and $\alpha_v = fK_v$. Equations (4.9) can be put in matrix form using homogeneous representation

$$\lambda \tilde{\mathbf{s}}_I = \lambda \begin{bmatrix} u_I \\ v_I \\ 1 \end{bmatrix} = \mathbf{K} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} \tag{4.10}$$

where

$$\mathbf{K} = \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \mathbf{\Omega\Pi} \tag{4.11}$$

is the matrix of *intrinsic parameters* $(\alpha_x, \alpha_y, u_0, v_0)$ and $\lambda = z_c > 0$ (The object is always in front of the camera). Using (4.5) yields

$$\begin{aligned} \lambda \tilde{\mathbf{s}}_I &= \mathbf{K}\mathbf{T}_b^c \tilde{\mathbf{p}}_b = \mathbf{W}\tilde{\mathbf{p}}_b \\ &= \begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} \\ w_{21} & w_{22} & w_{23} & w_{24} \\ w_{31} & w_{32} & w_{33} & w_{34} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} \alpha_u & 0 & u_0 & 0 \\ 0 & \alpha_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \end{aligned} \tag{4.12}$$

The Matrix $\mathbf{T}_b^c$ can be written as

$$\mathbf{T}_b^c = \begin{bmatrix} \mathbf{r}_1 & t_x \\ \mathbf{r}_2 & t_y \\ \mathbf{r}_3 & t_z \\ 0 & 1 \end{bmatrix} \tag{4.13}$$

where $\mathbf{r}_i = [r_{i1} \quad r_{i2} \quad r_{i3}]$ for $i = 1, 2, 3$. Matrix $\mathbf{W}$ can then be written in a more compact form

$$\mathbf{W} = \begin{bmatrix} \alpha_u \mathbf{r}_1 + u_0 \mathbf{r}_3 & \alpha_u t_x + u_0 t_z \\ \alpha_v \mathbf{r}_2 + v_0 \mathbf{r}_3 & \alpha_v t_y + v_0 t_z \\ \mathbf{r}_3 & t_z \end{bmatrix} \tag{4.14}$$

In practice, the axes of the pixel image frame $\{U_I, V_I\}$ are not perpendicular. To model this effect, the angle δ between the vectors $\mathbf{u}_I$ and $\mathbf{v}_I$, is introduced in the previous equations. A new image frame $u_I^{'}, v_I^{'}$ is then defined and the transformation between the two frames is

$$\begin{aligned} u_I^{'} &= u_I \cos\delta + v_I \sin\delta \\ v_I^{'} &= v_I \end{aligned} \tag{4.15}$$

and by defining

$$\begin{aligned} u_0^{'} &= u_0 \sin\delta + v_0 \cos\delta \\ v_0^{'} &= v_0 \end{aligned} \tag{4.16}$$

then the matrix of intrinsic parameters becomes

$$\mathbf{K} = \begin{bmatrix} \alpha & \gamma & u_0^{'} & 0 \\ 0 & \beta & v_0^{'} & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \tag{4.17}$$

where $\alpha = \alpha_u$, $\beta = \alpha_v / \sin\delta$ and $\gamma = \alpha_u \cot\delta$. The new global camera model is

$$\mathbf{W} = \begin{bmatrix} \alpha\mathbf{r}_1 + \gamma\mathbf{r}_2 + u_0^{'}\mathbf{r}_3 & \alpha t_x + u_0^{'} t_z \\ \beta\mathbf{r}_2 + v_0^{'}\mathbf{r}_3 & \beta t_y + v_0^{'} t_z \\ \mathbf{r}_3 & t_z \end{bmatrix} \tag{4.18}$$

4.2.2 Camera Calibration

Once a type of model has been chosen, its parameters must be identified. The estimated values of these parameters for a camera is performed by calibration. It is a necessary step for any vision-oriented application. Many techniques have been developed. They can be classified into two categories:

- **Calibration using calibration patterns**: This technique uses the observation of 3D objects with known coordinates. Calibration objects (calibration patterns) are generally distributed points on orthogonal planes or on a plane translated in the direction of its normal. The calculation can be done in a relatively simple way, e.g., see [100], [83], [48], [11], [110].

- **Automatic Calibration**: The known movement of the camera that is filming a static scenery is used to add constraints on the intrinsic parameters, taking into account the rigidity of objects filmed using only image information [35], [45].

4.2.2.1 Implementation of the Calibration

From the equation of global model (4.12), we obtain

$$\begin{aligned} u_I &= \frac{w_{11}x + w_{12}y + w_{13}z + w_{14}}{w_{31}x + w_{32}y + w_{33}z + w_{34}} \\ v_I &= \frac{w_{21}x + w_{22}y + w_{23}z + w_{24}}{w_{31}x + w_{32}y + w_{33}z + w_{34}} \end{aligned} \tag{4.19}$$

These equations are linear in the coefficients w_{ij}. Therefore, we need at least 12 equations of this kind to determine the 12 coefficients $\mathbf{W}$. In other words, we must know at least six points in the reference object and their respective projections on image. For n known points we obtain $2n$ equations of the form

$$\begin{aligned} w_{11}x_i + w_{12}y_i + w_{13}z_i + w_{14} - u_i w_{31}x_i - u_i w_{32}y_i - u_i w_{33}z_i = u_i w_{34} \\ w_{21}x_i + w_{22}y_i + w_{23}z_i + w_{24} - v_i w_{31}x_i - v_i w_{32}y_i - v_i w_{33}z_i = v_i w_{34} \end{aligned} \tag{4.20}$$

This system of equations can be put in matrix form

$$\begin{bmatrix} & & & & & & \vdots & & & & \\ x_i & y_i & z_i & 1 & 0 & 0 & 0 & 0 & -u_i x_i & -u_i y_i & -u_i z_i \\ 0 & 0 & 0 & 0 & x_i & y_i & z_i & 1 & -v_i x_i & -v_i y_i & -v_i z_i \\ & & & & & & \vdots & & & & \end{bmatrix} \begin{bmatrix} w_{11} \\ w_{12} \\ w_{13} \\ w_{14} \\ w_{21} \\ w_{22} \\ w_{23} \\ w_{24} \\ w_{31} \\ w_{32} \\ w_{33} \end{bmatrix} = \begin{bmatrix} \vdots \\ u_i w_{34} \\ v_i w_{34} \\ \vdots \end{bmatrix}$$

which we can write in a more compact form

$$\mathbf{A}\mathbf{x}_w = \mathbf{u} \tag{4.21}$$

where $\mathbf{x}_w$ is the unkown variable that needs to be identified.

Solving this equation gives us the values of w_{ij}. And using w_{ij} we can identify the intrinsic parameters through equations

$$\begin{aligned} u_0 &= \mathbf{w}_1^T \mathbf{w}_3 \\ v_0 &= \mathbf{w}_2^T \mathbf{w}_3 \\ \alpha_u &= \|\mathbf{w}_1 \times \mathbf{w}_3\| \\ \alpha_v &= \|\mathbf{w}_2 \times \mathbf{w}_3\| \end{aligned} \tag{4.22}$$

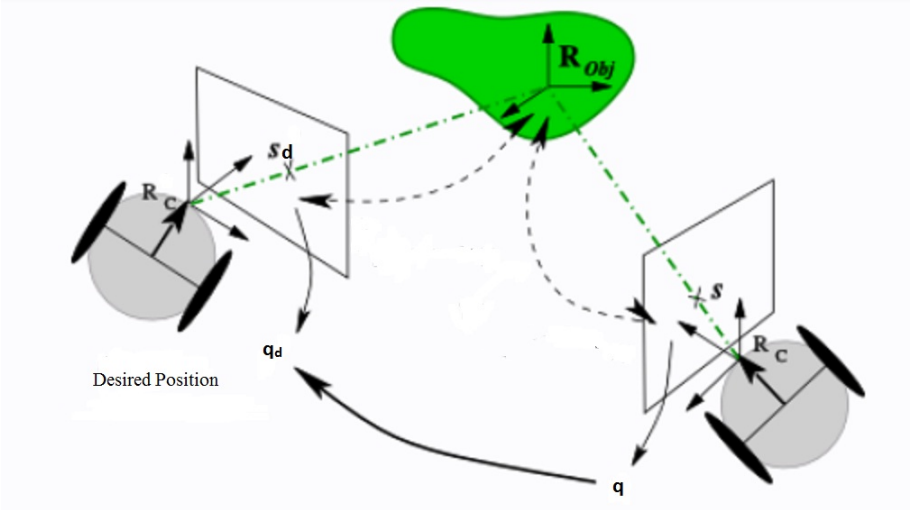


Figure 4.5: Position-based visual servoing (3D).

and the extrinsic parameters

$$\begin{aligned}
\mathbf{r}_3 &= \mathbf{w}_3 \\
\mathbf{r}_1 &= \frac{1}{\alpha_u}(\mathbf{w}_1 - u_0 \cdot \mathbf{w}_3) \\
\mathbf{r}_2 &= \frac{1}{\alpha_v}(\mathbf{w}_2 - v_0 \cdot \mathbf{w}_3) \\
t_x &= \frac{1}{\alpha_u}(w_{14} - u_0 \cdot w_{34}) \\
t_y &= \frac{1}{\alpha_v}(w_{24} - v_0 \cdot w_{34}) \\
t_z &= w_{34}
\end{aligned} \tag{4.23}$$

where $\mathbf{w}_i = [w_{i1} \;\; w_{i2} \;\; w_{i3}]$ for $i = 1, 2, 3$. So the problem of calibration is practically a problem of finding $\mathbf{W}$.

In order to calibrate the cameras, we adopted an algorithm based on Zhang's method [110], see Appendix C for more details.

4.3 Position-based Visual Servoing

Position-based visual servoing uses as an input of its control loop the three-dimensional information, namely the position and orientation of the robot relative to the object of interest. The task ahead is then expressed as a baseline to reach a desired posture

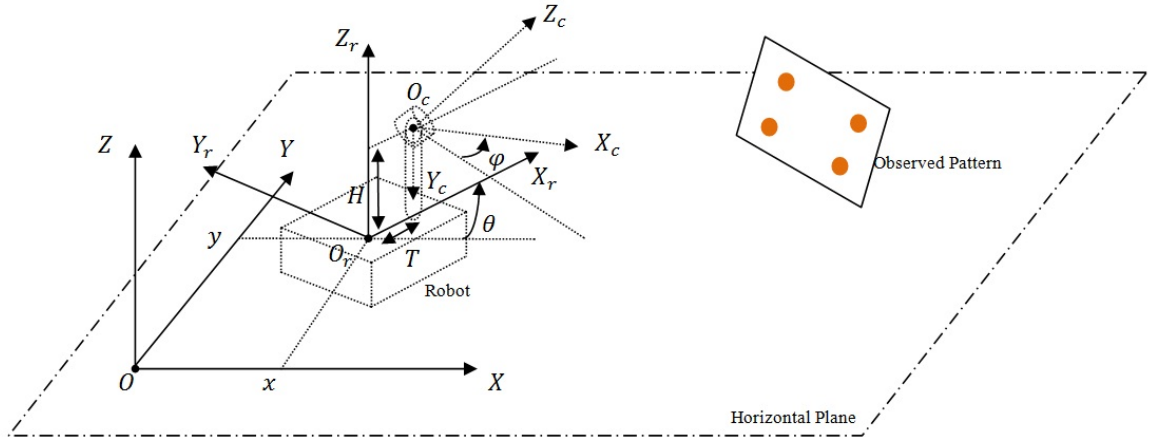


Figure 4.6: Robot and camera frames.

$\mathbf{q}_d = [x_d \quad y_d \quad \theta_d]$, or to track a trajectory $\mathbf{q}_d(t) = [x_d(t) \quad y_d(t) \quad \theta_d(t)]$. The control is thus based on the pose estimation, which is the determination of $\mathbf{q}$ of the robot based on visual information extracted from the image, and the camera in that case plays the role of a position sensor (Fig. 4.5).

Let's consider a pinhole camera mounted on a turret of height H, situated on the sagittal axis of the unicycle at a distance T from the wheels axis. Camera position relative to the robot is fixed, and it is capable of performing a rotation around its Y_c axis with an angle φ that can be measured using embedded sensor (Fig. 4.6).

Let $\tilde{\mathbf{p}}_b = [x_o \quad y_o \quad H_o \quad 1]_b^T$ a point of the observed object (pattern) expressed in the world frame using homogeneous representation, its projection in the image is $\tilde{\mathbf{p}}_I = [u_I \quad v_I \quad 1]^T$. Then, as we have seen, the relation $\lambda\tilde{\mathbf{p}}_I = \mathbf{K}\mathbf{T}_b^c\tilde{\mathbf{p}}_b$ holds. We assume that the camera has been calibrated to obtain the intrinsic parameters matrix $\mathbf{K}$, then the pose estimation problem is consists in finding $\mathbf{T}_b^c$. Equations (4.23) can be used, but since the robot is restricted to move in a horizontal plane we have four DOF of the camera instead of six, and since the camera rotation angle φ is known, we are left with three unknowns to be found (x, y and θ).

The relation between the point coordinates in world frame and its coordinates in camera frame is:

$$\tilde{\mathbf{p}}_c = \mathbf{T}_b^c\tilde{\mathbf{p}}_b = \mathbf{T}_r^c\mathbf{T}_b^r\tilde{\mathbf{p}}_b \tag{4.24}$$

By careful examination of Fig. 4.6 we can write

$$\mathbf{T}_r^b = \begin{bmatrix} \mathbf{R}_z(\theta) & \mathbf{o}_r^b \\ \mathbf{0} & 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & x \\ \sin\theta & \cos\theta & 0 & y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.25}$$

Hence

$$\mathbf{T}_b^r = (\mathbf{T}_r^b)^{-1} = \begin{bmatrix} \cos\theta & \sin\theta & 0 & -x\cos\theta - y\sin\theta \\ -\sin\theta & \cos\theta & 0 & x\sin\theta - y\cos\theta \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.26}$$

Also we have

$$\mathbf{T}_c^r = \begin{bmatrix} \mathbf{R}_z(\varphi)\mathbf{R}_x(-\frac{\pi}{2})\mathbf{R}_y(\frac{\pi}{2}) & \mathbf{o}_c^r \\ \mathbf{0} & 1 \end{bmatrix} = \begin{bmatrix} \sin\varphi & 0 & \cos\varphi & T \\ -\cos\varphi & 0 & \sin\varphi & 0 \\ 0 & -1 & 0 & H \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.27}$$

Hence

$$\mathbf{T}_r^c = (\mathbf{T}_c^r)^{-1} = \begin{bmatrix} \sin\varphi & -\cos\varphi & 0 & -T\sin\varphi \\ 0 & 0 & -1 & H \\ \cos\varphi & \sin\varphi & 0 & -T\cos\varphi \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.28}$$

Using (4.10) and (4.24) yields

$$\tilde{\mathbf{p}}_r = \mathbf{T}_c^r \tilde{\mathbf{p}}_c = \mathbf{T}_c^r z_c \mathbf{K}^{-1} \tilde{\mathbf{p}}_I = \mathbf{T}_b^r \tilde{\mathbf{p}}_b$$

$$\begin{bmatrix} \sin\varphi & 0 & \cos\varphi & T \\ -\cos\varphi & 0 & \sin\varphi & 0 \\ 0 & -1 & 0 & H \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{z_c}{\alpha_x}(u_I - u_0) \\ \frac{z_c}{\alpha_y}(v_I - v_0) \\ z_c \\ 1 \end{bmatrix}$$
$$= \begin{bmatrix} \cos\theta & \sin\theta & 0 & -x\cos\theta - y\sin\theta \\ -\sin\theta & \cos\theta & 0 & x\sin\theta - y\cos\theta \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_o \\ y_o \\ H_o \\ 1 \end{bmatrix} \tag{4.29}$$

The third line of this matrix equality yields

$$-\frac{z_c}{\alpha_y}(v_I - v_0) + H = H_o \tag{4.30}$$

$$z_c = \frac{\alpha_y(H - H_o)}{v_I - v_0} \tag{4.31}$$

Remark 6 *In order for this approach to be applicable, we must choose object points which have heights H_o that are different from the camera height H. The greater the value of $|H - H_o|$, the less the pose estimation is affected by measurements errors.*

However, if the object is on a vertical plane, then this shortcoming can be be overcome by considering two points of the objects $\boldsymbol{p}_{b1} = [x_o \;\; y_o \;\; H_{o1}]_b^T$ and $\boldsymbol{p}_{b2} = [x_o \;\; y_o \;\; H_{o2}]_b^T$, with $H_{o1} \neq H_{o2}$. Their projections in the image are $s_{I1} = [u_{I1} \;\; v_{I1}]$ and $s_{I2} = [u_{I2} \;\; v_{I2}]$ respectively. By assuming that the camera is sufficiently far from the object to consider that all the object's points have the same depth z_c, (4.30) yields

$$z_c = -\frac{\alpha_y(H_{o1} - H_{o2})}{v_{I1} - v_{I2}} \tag{4.32}$$

which is independent of H.

From equations (4.29) and (4.31), we finally obtain the target position in the robot frame

$$\begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix} = \begin{bmatrix} \frac{\alpha_y(H-H_o)}{v_I - v_0}\left(\frac{u_I - u_0}{\alpha_x}\sin\varphi + \cos\varphi\right) + T \\ \frac{\alpha_y(H-H_o)}{v_I - v_0}\left(-\frac{u_I - u_0}{\alpha_x}\cos\varphi + \sin\varphi\right) \\ H_o \end{bmatrix} \tag{4.33}$$

Equation (4.33) can be used to determine the relative position of the target in the robot frame, but to determine the orientation θ, at least two points from the object are required. The position of the target object in the robot frame is transformed into the world frame using (4.25) as follows

$$\begin{bmatrix} x_o \\ y_o \end{bmatrix} = \begin{bmatrix} x + x_r\cos\theta - y_r\sin\theta \\ y + x_r\sin\theta + y_r\cos\theta \end{bmatrix} \tag{4.34}$$

So by choosing two points from the target object (x_{o1}, y_{o1}) and (x_{o2}, y_{o2}), their coordinates in the robot frame, (x_{r1}, y_{r1}) and (x_{r2}, y_{r2}) respectively, can be obtained by (4.33). This leads to the following equation:

$$\begin{bmatrix} x_{o1} \\ y_{o1} \\ x_{o2} \\ y_{o2} \end{bmatrix} = \begin{bmatrix} 1 & 0 & x_{r1} & -y_{r1} \\ 0 & 1 & y_{r1} & x_{r1} \\ 1 & 0 & x_{r2} & -y_{r2} \\ 0 & 1 & y_{r2} & x_{r2} \end{bmatrix} \begin{bmatrix} x \\ y \\ C_\theta \\ S_\theta \end{bmatrix} = \mathbf{H}\bar{\mathbf{q}} \tag{4.35}$$

where $C_\theta = \cos\theta$ and $S_\theta = \sin\theta$. When $\det(\mathbf{H}) = (x_{r1} - x_{r2})^2 + (y_{r1} - y_{r2})^2 \neq 0$ (which is always true except in the trivial case when the two points are identical), equation (4.35) can be used to obtain x, y and θ.

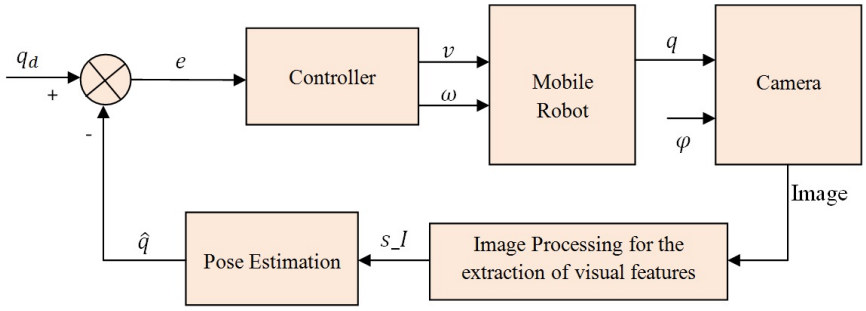


Figure 4.7: Block diagram of position-based visual servoing.

Now because of the noise affecting the measurements of the coordinates in the image plane, the results of this method are affected by errors. To make the pose estimation more robust, a number of points $n > 2$ are considered and $\bar{\mathbf{q}}$ is computed using least-squares techniques. This, however, does not guarantee that the resulting C_θ and S_θ are valid (cos) and (sin) values, so the following values are used instead

$$\cos\theta = \frac{C_\theta}{\sqrt{C_\theta^2+S_\theta^2}} \tag{4.36}$$

$$\sin\theta = \frac{S_\theta}{\sqrt{C_\theta^2+S_\theta^2}} \tag{4.37}$$

Once $\mathbf{q}$ has been determined using image data, the problem becomes a pure motion control problem discussed in chapter 3, and control block diagram is shown in Fig. 4.7.

Remark 7 *The camera's rotation angle φ is independent of the control loop and its value is known at any given time, it can be used to guarantee that the observed object stays in the field of view at all times.*

4.4 Image-based Visual Servoing

Image-based visual servoing (2D) techniques directly use the visual information, denoted $\mathbf{s}$, extracted from the image. The task ahead then is specified directly in the image in terms of the desired visual feature $\mathbf{s}_d$ to achieve. The control law is then to control the camera movement so as to cancel the error between the current visual information $\mathbf{s}(t)$ and $\mathbf{s}_d$ the desired pattern (see Fig. 4.8). This approach doesn't require the 3D reconstruction of the target.

b

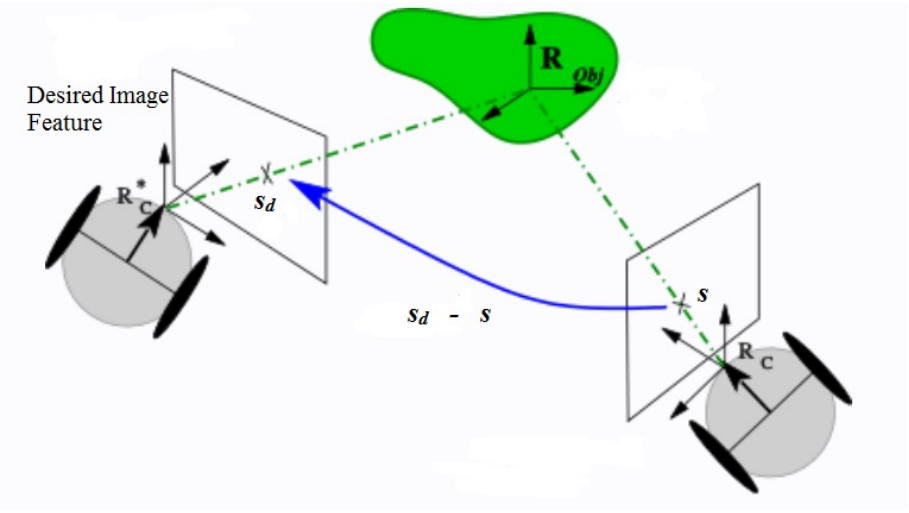


Figure 4.8: Image-based visual servoing (2D).

Usually, $\mathbf{s}$ is composed of the image coordinates of several points belonging to the considered object (Target). However, other image features can be considered like image moments as we will see. As for $\mathbf{s}_d$, it is obtained either during an off-line learning step (where the robot is moved at its desired position with respect to the target object and the corresponding image is acquired), or by computing the projection in the image of a 3D model of the target for the desired camera pose (which necessitates a perfect camera calibration and a perfect knowledge of the 3D target model).

Let's define the image feature vector $\mathbf{s}(\mathbf{m}(t), a)$. The vector $\mathbf{m}(t)$ is a set of image measurements (e.g., the image coordinates of interest points, or the parameters of a set of image moments). These image measurements are used to compute a vector of k visual features. a is a set of parameters that represent potential additional knowledge about the system (e.g., coarse camera intrinsic parameters or three-dimensional model of objects). For example, if the image features are points, then we have

$$\mathbf{s} = [u_{I1} \quad v_{I1} \quad u_{I2} \quad v_{I2} \cdots u_{Ik} \quad v_{Ik}]^T \tag{4.38}$$

where $(u_{Ii} \quad v_{Ii}) \quad i = 1 \cdots k$ are pixel coordinates of these points. The vector $\mathbf{s}_d$ contains the desired values of the features. If the camera is in motion with respect to the object (or the other way around), the feature vector $\mathbf{s}$ is, in general, time-varying. The

velocity vector of the camera is the same as the robot velocity vector $\mathbf{v} = [v \;\; \omega]_r^T$. The relationship between $\dot{\mathbf{s}}$ and $\mathbf{v}$ is given by

$$\dot{\mathbf{s}} = \mathbf{J}_s(\mathbf{s}, \mathbf{T}_b^c)\mathbf{v} \tag{4.39}$$

where $\mathbf{J}_s$ is a $(2k \times 2)$ matrix termed *Image Jacobian*[1]. This equation is linear but $\mathbf{J}_s$ depends, in general, on the current value of the feature vector $\mathbf{s}$ and on the relative pose of the object with respect to the camera T_b^c.

The aim of all image-based control schemes is to minimize an error $\mathbf{e}_s(t)$, which is typically defined by

$$\mathbf{e}_s = \mathbf{s}_d - \mathbf{s} \tag{4.40}$$

Using (4.39) and (4.40) we immediately obtain the relationship between the camera velocity and the time variation of the error

$$\dot{\mathbf{e}}_s = -\mathbf{J}_e\mathbf{v} \tag{4.41}$$

where $\mathbf{J}_e = \mathbf{J}_s$.

Considering $\mathbf{v}$ as the input to the robot system, we want to ensure an exponential decoupled decrease of the error

$$\dot{\mathbf{e}}_s = -\mathbf{K}_s\mathbf{e}_s \tag{4.42}$$

Therefore, if $\mathbf{K}_s$ is a positive definite matrix, system (4.42) is asymptotically stable and $\mathbf{e}_s$ tends to zero with exponential convergence and the rate of convergence depends on the eigenvalues of matrix $\mathbf{K}_s$.

Using 4.41 and 4.42, we obtain

$$\mathbf{v} = \widehat{\mathbf{J}_s^+}\mathbf{K}_s\mathbf{e}_s \tag{4.43}$$

where $\widehat{\mathbf{J}_s^+} \in \mathbb{R}^{(2\times k)}$ is an estimation of the Moore-Penrose pseudo-inverse of $\mathbf{J}_s$, that is [80], $\mathbf{J}_s^+ = (\mathbf{J}_s^T\mathbf{J}_s)^{-1}\mathbf{J}_s^T$, given that $\mathbf{J}_s$ is of full rank 2.

A sufficient condition to ensure the global asymptotic stability of the system is (see [49])

$$\widehat{\mathbf{J}_s^+}\mathbf{J}_s(\mathbf{s}(t)) > 0 \qquad \forall t \tag{4.44}$$

In the literature, three different choices for $\mathbf{J}_s^+$ have been considered [21]:

[1]The term *Interaction Matrix* is also used interchangeably with *Image Jacobian* in the visual servoing literature.

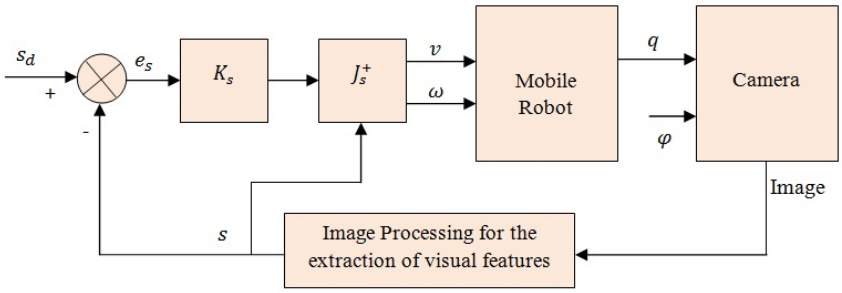


Figure 4.9: Block diagram of image-based visual servoing.

- $\widehat{\mathbf{J}_s^+} = \widehat{\mathbf{J}_s^+}(t)$. In that case, the image Jacobian is numerically estimated during the camera motion without taking into account the analytical form of the Image Jacobian. This approach seems to be very interesting if any camera and robot models are available. However, it is impossible in that case to demonstrate whether condition (4.44) is ensured. Furthermore, initial coarse estimation of the image Jacobian may lead to unstable results, especially at the beginning of the servoing, and some visual features may get out of the camera field of view.

- $\widehat{\mathbf{J}_s^+} = \mathbf{J}_s^+(\mathbf{s}_d)$. In this last case, $\widehat{\mathbf{J}_s^+}$ is constant and determined during an off-line step using the desired value of the visual features. Condition (4.44) is now ensured only in a neighborhood of the desired position, and a decoupled behavior will be achieved only in a smaller neighborhood. The performed trajectory in the image may be unpredictable, and some visual features may get out of the camera field of view during the servoing, especially if the initial camera position is far away from its desired one.

- $\widehat{\mathbf{J}_s^+} = \mathbf{J}_s^+(\mathbf{s}(t))$. The image Jacobian is now updated at each iteration of the control law using the current measurement of the visual features $\mathbf{s}(t)$. In this case we have $\widehat{\mathbf{J}_s^+}\mathbf{J}_s(\mathbf{s}(t)) = \mathbb{I}, \forall t$, which satisfies condition (4.44) and implies a perfect decoupled system. Each image point is constrained to reach its desired position following a straight line. However, this may result in an inadequate robot motion.

In our approach we will use the choice $\widehat{\mathbf{J}_s^+} = \mathbf{J}_s^+(s(t))$ as shown in Fig. 4.9. To this end, we need to find the analytical form of the Image Jacobian for different kinds of

image features.

4.4.1 Image Jacobian of a point

Let $\mathbf{p}_b = [x_o \;\; y_o \;\; H_o]_b^T$ a point of the observed object (pattern) expressed in the world frame, its projection in the image is $\mathbf{p}_I = [u_I \;\; v_I]^T$. In the following, to simplify notation, normalized coordinates (X, Y) will be used in place of pixel coordinates (u_I, v_I) to define the feature vector.

$$\begin{aligned} X =& \frac{1}{\alpha_x}(u_I - u_0) \\ Y =& \frac{1}{\alpha_y}(v_I - v_0) \end{aligned} \tag{4.45}$$

And the vector of image features is thus $\mathbf{s} = [X \;\; Y]^T$.

Since only pixel coordinates can be directly measured, the normalized coordinates should be computed from pixel coordinates using (4.45), provided that the intrinsic parameters of the camera are known.

Let $\mathbf{p}_c = [x_c \;\; y_c \;\; z_c]_c^T$ be the vector of coordinates of the point $\mathbf{p}_b$ expressed in the camera frame, and using (4.10) and (4.45) yields

$$\mathbf{s}(\mathbf{p}_c) = \begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} x_c/z_c \\ y_c/z_c \end{bmatrix} \tag{4.46}$$

Computing the time derivative of (4.46) yields

$$\dot{\mathbf{s}} = \frac{\partial \mathbf{s}(\mathbf{p}_c)}{\partial \mathbf{p}_c} \dot{\mathbf{p}}_c \tag{4.47}$$

by using (4.46), we have

$$\frac{\partial \mathbf{s}(\mathbf{p}_c)}{\partial \mathbf{p}_c} = \frac{1}{z_c} \begin{bmatrix} 1 & 0 & -X \\ 0 & 1 & -Y \end{bmatrix} \tag{4.48}$$

And using homogeneous representation we have

$$\begin{aligned} \tilde{\mathbf{p}}_c &= \mathbf{T}_r^c \mathbf{T}_b^r \tilde{\mathbf{p}}_b \\ &= \begin{bmatrix} \sin(\theta+\varphi) & -\cos(\theta+\varphi) & 0 & -x\sin(\theta+\varphi) + y\cos(\theta+\varphi) - T\sin(\varphi) \\ 0 & 0 & -1 & H \\ \cos(\theta+\varphi) & \sin(\theta+\varphi) & 0 & -x\cos(\theta+\varphi) - y\sin(\theta+\varphi) - T\cos(\varphi) \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ H_0 \\ 1 \end{bmatrix} \end{aligned} \tag{4.49}$$

By computing the time derivative of (4.49) and using the kinematic model (1.3) of the mobile robot, we obtain

$$\dot{\mathbf{p}}_c = \begin{bmatrix} \dot{x}_c \\ \dot{y}_c \\ \dot{z}_c \end{bmatrix} = \begin{bmatrix} -\sin\varphi & z_c + T\cos\varphi \\ 0 & 0 \\ -\cos\varphi & -x_c - T\sin\varphi \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{4.50}$$

By substituting (4.48) and (4.50) in (4.47), we obtain the image Jacobian matrix of a point

$$\mathbf{J}_s(\mathbf{s},\varphi) = \begin{bmatrix} \frac{1}{z_c}(-\sin\varphi + X\cos\varphi) & 1 + X^2 + \frac{T}{z_c}(\cos\varphi + X\sin\varphi) \\ \frac{1}{z_c}Y\cos\varphi & XY + \frac{T}{z_c}Y\sin\varphi \end{bmatrix} \tag{4.51}$$

From (4.31), we can write z_c as a function of Y

$$z_c = \frac{H - H_o}{Y} \tag{4.52}$$

Substituting in (4.53) yields

$$\mathbf{J}_s(\mathbf{s},\varphi) = \begin{bmatrix} \frac{1}{H-H_o}(-Y\sin\varphi + XY\cos\varphi) & 1 + X^2 + \frac{T}{H-H_o}(Y\cos\varphi + XY\sin\varphi) \\ \frac{1}{H-H_o}Y^2\cos\varphi & XY + \frac{T}{H-H_o}Y^2\sin\varphi \end{bmatrix} \tag{4.53}$$

Remark 8 *In the case where the camera is parallel to the sagittal axis of the mobile robot and facing objects in front of the robot ($\varphi = 0$), the Image Jacobian becomes*

$$\mathbf{J}_s(\boldsymbol{s}) = \begin{bmatrix} \frac{XY}{H-H_o} & 1 + X^2 + \frac{T}{H-H_o}Y \\ \frac{1}{H-H_o}Y^2 & XY \end{bmatrix} \tag{4.54}$$

4.4.2 Image Jacobian of a set of points

The image Jacobian matrix of a set of k points of the object $\mathbf{p}_1, \cdots \mathbf{p}_k$ can be built by considering the $(2k \times 1)$ feature vector (4.38). If $\mathbf{J}_{\mathbf{s}_i}(\mathbf{s}_i, \varphi)$ denotes the image Jacobian matrix corresponding to point $\mathbf{p}_i$, then the image Jacobian matrix of the set of points will be the $(2k \times 2)$ matrix

$$\mathbf{J}_s(\mathbf{s},\varphi) = \begin{bmatrix} \mathbf{J}_{s1}(\mathbf{s}_1,\varphi) \\ \vdots \\ \mathbf{J}_{sk}(\mathbf{s}_k,\varphi) \end{bmatrix} \tag{4.55}$$

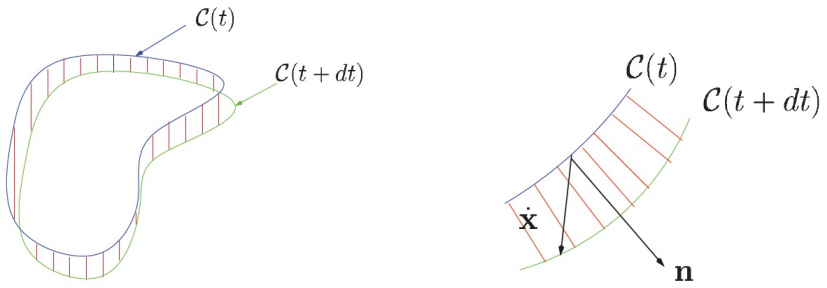


Figure 4.10: Time variation of contour $\mathcal{C}(t)$.

4.4.3 Image Jacobian of image moments

The objective here is to find the analytic form of the image Jacobian that links the derivative of an image moment $m_{i,j}$ of an observed object in the image, with the vector $\mathbf{v} = [v \quad \omega]^T$. In other words we will obtain a linear link that can be expressed under the form

$$\dot{m}_{i,j} = \mathbf{J}_{mij}\mathbf{v} \tag{4.56}$$

Let $\mathcal{O}$ be the observed object and $i(t)$ the image acquired by the camera at time t. We denote by $\mathcal{R}(t)$ the part of $i(t)$ where the object projects, and $\mathcal{C}(t)$ the contour of $\mathcal{R}(t)$. We will consider here that either binary images are acquired or a spatial segmentation algorithm, providing binary images, is first performed on the acquired images. In that case, the moments $m_{i,j}$ of $\mathcal{O}$ in the image are defined by [23]

$$m_{i,j}(t) = \iint_{\mathcal{R}(t)} f(X,Y)dXdY \tag{4.57}$$

where $f(X,Y) = X^iY^j$.

In (4.57), the only part that is a function of time is $\mathcal{R}(t)$. The time variation of m_{ij} can thus be obtained from the variation of $\mathcal{C}(t)$. More precisely, we have (see Fig. 4.10)

$$\dot{m}_{i,j}(t) = \oint_{\mathcal{C}(t)} f(X,Y)\dot{\mathbf{x}}^T \cdot \mathbf{n}dl \tag{4.58}$$

where $\dot{\mathbf{x}}$ is the velocity of contour point $\mathbf{x} = [X \quad Y]^T$, $\mathbf{n}$ is the unitary vector normal to $\mathcal{C}(t)$ at point $\mathbf{x}$, and dl is an infinitesimal element of contour $\mathcal{C}(t)$. If the following conditions are satisfied (which is the case in practice):

1. $\mathcal{C}(t)$ is continuous by parts.

2. vector $f(X,Y)\cdot\dot{\mathbf{x}}^T$ is tangent to $\mathcal{R}(t)$ and continuously differentiable.

then using Green's theorem (Appendix B.1) yields

$$\oint_{\mathcal{C}(t)} f(X,Y)\dot{\mathbf{x}}^T\cdot\mathbf{n}dl = \iint_{\mathcal{R}(t)} \text{div}\ \left[f(X,Y)\dot{\mathbf{x}}\right] dXdY \tag{4.59}$$

By developing (4.59) and substituting the result in (4.58), we finally obtain

$$\dot{m}_{i,j}(t) = \iint_{\mathcal{R}(t)} \left[\frac{\partial f}{\partial X}\dot{X} + \frac{\partial f}{\partial Y}\dot{Y} + f(X,Y)(\frac{\partial \dot{X}}{\partial X} + \frac{\partial \dot{Y}}{\partial Y})\right] dXdY \tag{4.60}$$

Now using the expression of image Jacobian matrix of a point given in (4.51), we obtain the derivatives of point's coordinates (X,Y) in the image as functions of (v,ω)

$$\begin{aligned}
\dot{X} &= \left[\frac{1}{H-H_o}(-Y\sin\varphi + XY\cos\varphi)\right]v + \left[1+X^2+\frac{T}{H-H_o}(Y\cos\varphi + XY\sin\varphi)\right]\omega \\
\dot{Y} &= \left[\frac{1}{H-H_o}Y^2\cos\varphi\right]v + \left[XY+\frac{T}{H-H_o}Y^2\sin\varphi\right]\omega
\end{aligned} \tag{4.61}$$

Hence

$$\begin{aligned}
\frac{\partial \dot{X}}{\partial X} &= \left[\frac{Y}{H-H_o}\cos\varphi\right]v + \left[2X+\frac{T}{H-H_o}Y\sin\varphi\right]\omega \\
\frac{\partial \dot{Y}}{\partial Y} &= \left[\frac{2Y}{H-H_o}\cos\varphi\right]v + \left[X+\frac{2T}{H-H_o}Y\sin\varphi\right]\omega
\end{aligned} \tag{4.62}$$

And we have

$$\begin{aligned}
f(X,Y) &= X^iY^j \\
\frac{\partial f}{\partial X}(X,Y) &= iX^{i-1}Y^j \\
\frac{\partial f}{\partial Y}(X,Y) &= jX^iY^{j-1}
\end{aligned} \tag{4.63}$$

Now we consider the object planar and situated on a vertical plane, in this case we can simplify the integration by replacing $H-H_o$ with $H_m = H-H_g$, where H_g is the height of the center of gravity of the object, and we assume that the object is chosen such that $H_m \neq 0$ (see Remark 6). By substituting (4.61), (4.62) and (4.63) in (4.60), we obtain after development

$$\dot{m}_{i,j} = \mathbf{J}_{mij}\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} J^v_{mij} & J^\omega_{mij} \end{bmatrix}\begin{bmatrix} v \\ \omega \end{bmatrix} \tag{4.64}$$

where

$$\begin{aligned} J^{v}_{mij} &= \frac{1}{H_m}((i+j+3)m_{i,j+1}\cos\varphi - im_{i,j}\sin\varphi) \\ J^{\omega}_{mij} &= im_{i-1,j} + (i+j+3)m_{i+1,j} + \\ &\quad \frac{T}{H_m}\left[im_{i-1,j+1}\cos\varphi + (i+j+3)\, m_{i,j+1}\sin\varphi\right] \end{aligned} \tag{4.65}$$

4.4.4 Pose estimation algorithm based on the image Jacobian

The Image Jacobian matrix can be used in pose estimation, i.e. computing $\mathbf{q} = [x \quad y \quad \theta]^T$. As we have seen, the relation between the derivative of the image feature vector $\mathbf{s} = [X \quad Y]^T$ and the robot's velocities vector $\mathbf{v}$ is $\dot{\mathbf{s}} = \mathbf{J}_s\mathbf{v}$. By using the kinematic model of the mobile robot $\dot{\mathbf{q}} = \mathbf{G}(\mathbf{q})\mathbf{v}$, we obtain

$$\dot{\mathbf{s}} = \mathbf{J}_s(\mathbf{s})\mathbf{G}^{+}(\mathbf{q})\dot{\mathbf{q}} = \mathbf{A}_s(\mathbf{s},\mathbf{q})\dot{\mathbf{q}} \tag{4.66}$$

where $\mathbf{G}^{+}(\mathbf{q}) = (\mathbf{G}^T\mathbf{G})^{-1}\mathbf{G}^T$ is the pseudo-inverse of matrix $\mathbf{G}(\mathbf{q})$.

Equation (4.66) is the starting point of a numerical integration algorithm for the computation of $\mathbf{q}$. Let $\widehat{\mathbf{q}}$ denote the current estimate of vector $\mathbf{q}$ and let

$$\widehat{\mathbf{s}} = \mathbf{s}(\widehat{\mathbf{q}}) \tag{4.67}$$

be the corresponding vector of image feature parameters computed from the pose specified by $\widehat{\mathbf{q}}$; the objective of this algorithm is the minimization of the error

$$\mathbf{e}_s = \mathbf{s} - \widehat{\mathbf{s}} \tag{4.68}$$

Notice that, for the purpose of numerical integration, vector $\mathbf{s}$ is constant while the current estimate $\widehat{\mathbf{s}}$ depends on the current integration time. Therefore, computing the time derivative of (4.68) yields

$$\dot{\mathbf{e}}_s = -\dot{\widehat{\mathbf{s}}} = -\mathbf{A}_s(\widehat{\mathbf{s}},\widehat{\mathbf{q}})\dot{\widehat{\mathbf{q}}} \tag{4.69}$$

Assumming that the matrix $\mathbf{A}_s^{+}(\widehat{\mathbf{s}},\widehat{\mathbf{q}}) = (\mathbf{A}_s^T\mathbf{A}_s)^{-1}\mathbf{A}_s^T$ is nonsingular, the choice

$$\dot{\widehat{\mathbf{q}}} = \mathbf{A}_s^{+}(\widehat{\mathbf{s}},\widehat{\mathbf{q}})\mathbf{K}_s\mathbf{e}_s \tag{4.70}$$

leads to the equivalent linear system

$$\dot{\mathbf{e}}_s + \mathbf{K}_s\mathbf{e}_s = 0 \tag{4.71}$$

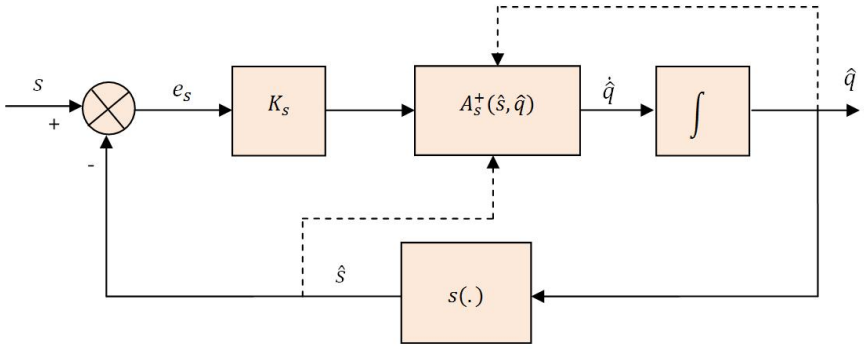


Figure 4.11: Pose estimation algorithm based on the image Jacobian.

Therefore, if $\mathbf{K}_s$ is a positive definite matrix (usually diagonal), the system (4.71) is asymptotically stable and the error tends to zero with a convergence rate that depends on the eigenvalues of matrix $\mathbf{K}_s$. The convergence to zero of error e_s ensures the asymptotic convergence of the estimate $\widehat{\mathbf{q}}$ toward the actual value $\mathbf{q}$.

The block scheme of the pose estimation algorithm is shown in Fig. 4.11, where $s(\cdot)$ denotes the function computing the feature vector of the 'virtual' image corresponding to the current estimate $\widehat{\mathbf{q}}$ of the pose. This algorithm can be used as an alternative to the analytic methods for pose estimation illustrated in Section 4.3. Obviously, the convergence properties depend on the choice of the image feature parameters and on the initial value of estimate $\widehat{\mathbf{q}}(0)$, which may produce instability problems related to the singularities of matrix A_s.

Notice that the pose estimation methods based on Jacobian are as efficient, in terms of accuracy, speed of convergence and computational load, as the initial estimate $\widehat{\mathbf{q}}(0)$ is close to the true value $\mathbf{q}$. Therefore, these methods are mainly adopted for real-time 'visual tracking' applications, where the estimate on an image taken at time t is computed assuming as initial value the estimate computed on the image taken at time $t - T$, T being the sampling time of the image.

4.5 Trajectory Planning on the Image Plane

Our approach for image-based visual servoing is based on the regulation to zero of an error function computed from the current measurement and a constant desired one. It

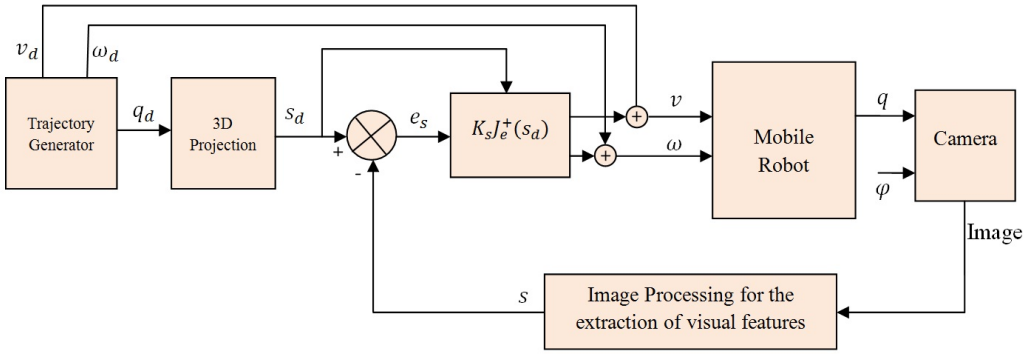


Figure 4.12: Block diagram of IBVS coupled with trajectory planning in the image plane.

is, therefore, not trivial to introduce any constraint in the realized trajectories such as the target object remains in the camera field of view, or to ensure the convergence of all the initial configurations. Moreover, we have used the approach $\widehat{J_s^+} = J_s^+(s(t))$ that requires the calculation of $J_s^+(s(t))$ at each iteration, which is time consuming and makes the matrix J_s, and hence the system, more vulnerable to image noise and measurements errors. On the other hand, it seems useful to use the approach $\widehat{J_s^+} = J_s^+(s_d)$ which can be calculated off-line. This, however, implies that the stability is ensured in a small neighborhood of $\mathbf{s}_d$.

One way to go around this problem is to perform a trajectory planning on the image plane, i.e. to provide desired image features values $\mathbf{s}_d(t)$ that corresponds to a feasible Cartesian trajectory. In other words, we will make the desired features acting as if the target object is observed by a 'virtual' mobile robot that moves from the initial position to its final destination following a desired path, as those described in chapter 2 for example, and our objective is to follow that robot. By this coupling of path planning in image space and image-based control, constraints such that the object remains in the camera field of view can be taken into account at the planning level. Furthermore, current measurements always remain close to their desired value and robustness of the image-based servoing is ensured along the whole trajectory.

As we have seen in section 4.4, the aim of image-based control scheme is to minimize the error

$$\mathbf{e}_s(t) = \mathbf{s}_d(t) - \mathbf{s}(t) \tag{4.72}$$

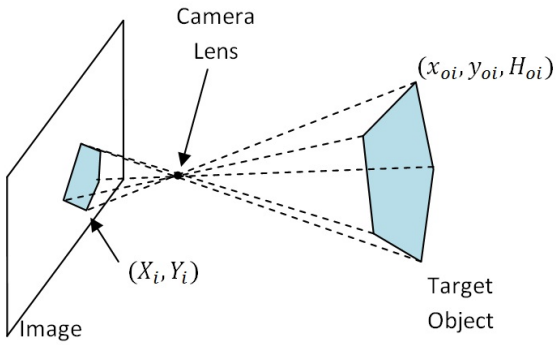


Figure 4.13: Polygon projection on the image plane.

Its derivative is then given by

$$\begin{aligned} \dot{\mathbf{e}}_s &= \mathbf{J}_s(\mathbf{s}_d(t))\mathbf{v}_d - \mathbf{J}_s(\mathbf{s}(t))\mathbf{v} \\ &\approx \mathbf{J}_s(\mathbf{s}_d(t))(\mathbf{v}_d - \mathbf{v}) \end{aligned} \tag{4.73}$$

where $\mathbf{v}_d$ is the velocity vector of the 'virtual' mobile robot. Using (4.42) the control law is then

$$\mathbf{v} = \mathbf{v}_d + \mathbf{J}_s^+(\mathbf{s}_d(t))\mathbf{K}_s\mathbf{e}_s \tag{4.74}$$

The values of $\mathbf{J}_s^+$ are calculated off-line along the trajectory using $\mathbf{s}_d$ values, which may be obtained during an off-line learning step (where the robot is moved at its desired position with respect to the target object and the corresponding image is acquired). This method, however, depends on position estimation using encoders informations (which may be unavailable or unreliable, and this is the reason why we resort to visual servoing in the first place), also, this method requires that the target object remains fixed throughout the learning phase. For these reasons, we propose to calculate the desired image features value by computing the projection in the image of a 3D model of the target for the desired camera pose. These values are then stored in the memory in a lookup table ready to be used when needed. Firstly, this approach requires no learning phase to be performed before the actual servoing phase, and secondly, target object movement can be easily incorporated into our equations.

When points are used as visual features, then the desired points locations are obtained by simple 3D projection using equation (4.12). Obtaining the desired visual

features, however, becomes more difficult when image moments are used as visual features. To perform a trajectory planning in the image plane using image moments we will consider two special cases:

- The target object is a planar polygon. Let (x_{o1}, y_{o1}, H_{o1}),(x_{o2}, y_{o2}, H_{o2}) $\cdots$ $(x_{ok}$, y_{ok}, $H_{ok})$ denote the vertices of the polygon expressed in world frame. Under the hypothesis that the distortion in the camera is very small, the projection of this polygon in the image is another polygon whose vertices (X_1, Y_1),(X_2, Y_2) $\cdots (X_k, Y_k)$ are the projections of the original polygon vertices on the image plane (normalized coordinates are used here) using equation (4.12) (see Fig. 4.13). The relationship between image moments of the polygon and its vertices can be then derived using Green's Theorem (see Appendix B.3). For example, the four first image moments are given by

$$
\begin{aligned}
m_{0,0} &= \frac{1}{2}\sum_{i=1}^{k}\left(X_i Y_{i+1} - X_{i+1}Y_i\right) \\
m_{1,0} &= \frac{1}{6}\sum_{i=1}^{k}\left(X_i + X_{i+1}\right)\left(X_i Y_{i+1} - X_{i+1}Y_i\right) \\
m_{0,1} &= \frac{1}{6}\sum_{i=1}^{k}\left(Y_i + Y_{i+1}\right)\left(X_i Y_{i+1} - X_{i+1}Y_i\right) \\
m_{1,1} &= \frac{1}{24}\sum_{i=1}^{k}\left[X_i^2 Y_{i+1}\left(2Y_i + Y_{i+1}\right) - X_{i+1}^2 Y_i\left(Y_i + 2Y_{i+1}\right) + 2X_i X_{i+1}\left(Y_{i+1}^2 - Y_i^2\right)\right]
\end{aligned}
\tag{4.75}
$$

  with $(X_{k+1}, Y_{k+1}) = (X_0, Y_0)$. Higher order moments can be derived in a similar manner (see Appendix B.3).

- The target object is a planar ellipse. In this case, image projection is also an ellipse whose antipodal points on its major axis and minor axis denoted (X_a, Y_a), (X_{-a}, Y_{-a}), (X_b, Y_b) and (X_{-b}, Y_{-b}) are the projections of their counterparts of the original object using equation (4.12). These four points can be then used to calculate the ellipse's major diameter $2a$, its minor diameter $2b$, the angle between its major axis and horizon δ and its center (X_g, Y_g) (see Fig. 4.14).

  The relationship between image moments of the ellipse and its elements can be

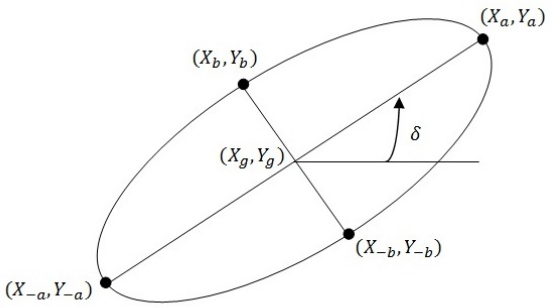


Figure 4.14: Usual representation of an ellipse.

then derived (see Appendix B.4):

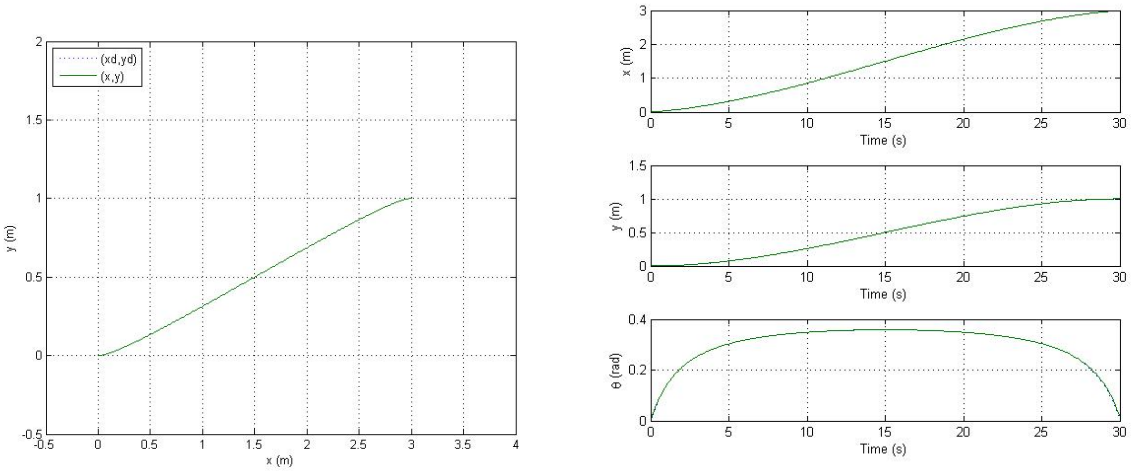
$$\begin{aligned} m_{0,0} &= \pi ab \\ m_{1,0} &= X_g m_{0,0} \\ m_{0,1} &= Y_g m_{0,0} \\ m_{1,1} &= X_g Y_g m_{0,0} + \frac{\pi}{8} ab(a^2 - b^2) \sin 2\delta \end{aligned} \tag{4.76}$$

4.6 Simulations Results

Position-based Visual Servoing

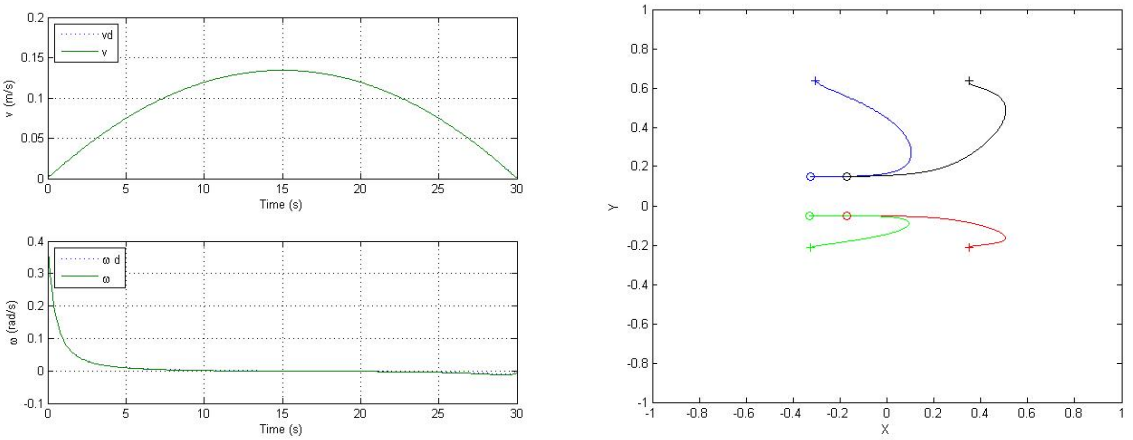
In the simulations we consider the normalized image coordinates; which are camera independent. Image features considered are four points (could be for example vertices of a polygon or centers of circles), their coordinates in world frame are $p_{o1} = (4m, 0.68m, 0.2m)$, $p_{o2} = (4m, 1.32m, 0.2m)$, $p_{w3} = (4m, 1.32m, 1.0m)$ and $p_{w4} = (4m, 0.68m, 1.0m)$. An analytical pose estimation (described in section 4.3) is used to determine the robot's position based on the image features for camera's height of $H = 0.4m$. To complete the control loop we used a nonlinear controller based on the I&I methodology described in section 3.4.3, to track a Cartesian trajectory that is designed using cubic polynomials that ensures that the target object remains in the camera's field of views, which is considered here to be $Width = -1 \cdots 1, Height = -1 \cdots 1$[1].

[1]Keeping in mind that we are using normalized coordinates for simulation purposes



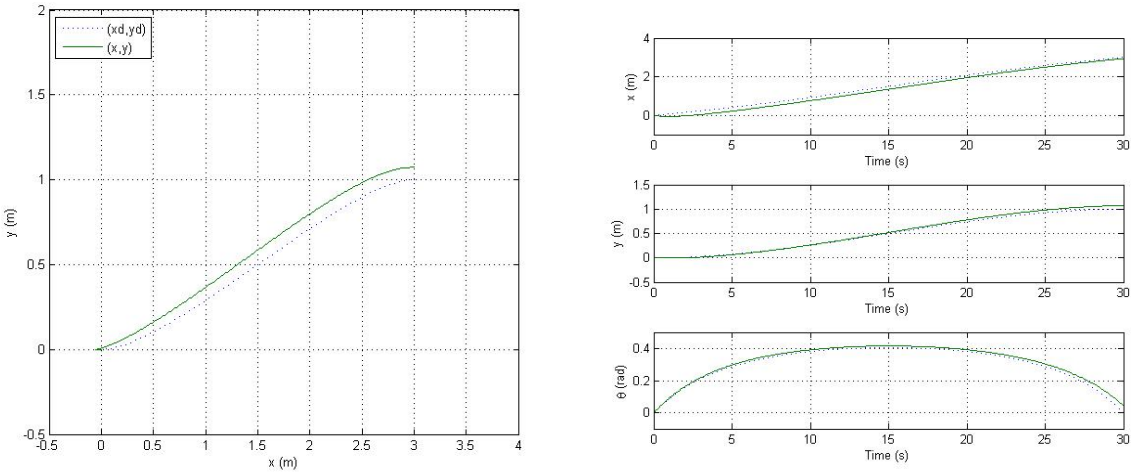
(a) Trajectory **(b)** Time Responses

Figure 4.15: Position-based visual servoing of robot system in tracking a trajectory planned via cubic Cartesian polynomials for a parking maneuver.



(a) Velocities **(b)** Image

Figure 4.16: Position-based visual servoing of robot system in tracking a trajectory planned via cubic Cartesian polynomials for a parking maneuver.



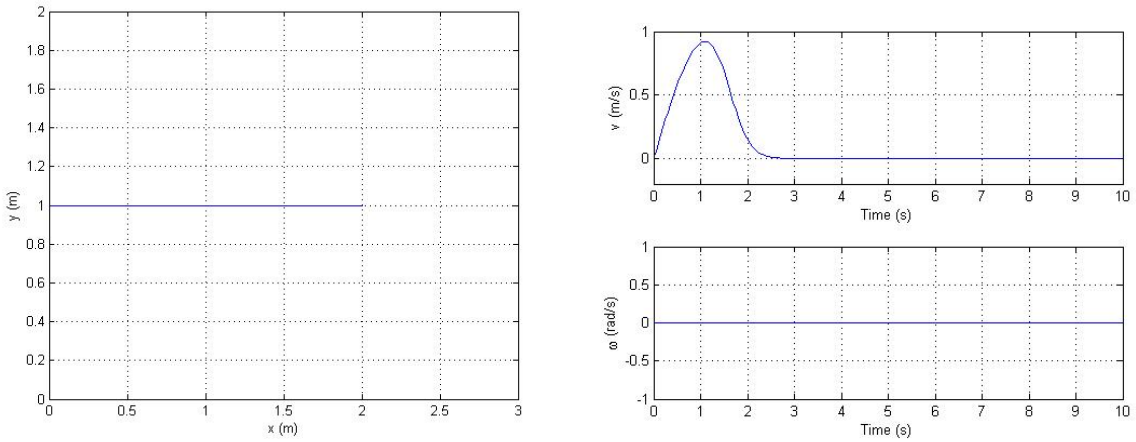
(a) Trajectory **(b)** Time Responses

Figure 4.17: Position-based visual servoing of robot system in tracking a trajectory planned via cubic Cartesian polynomials for a parking maneuver in the presence of calibration error.

Simulations have been performed in Simulink with a fixed step of $50ms$. Fig. 4.15 shows the tracking of a trajectory planned via cubic polynomials between the initial posture (pose) $(x_0, y_0, \theta_0) = (0, 0, 0)$ and the final posture $(x_f, y_f, \theta_f) = (3m, 1m, \pi/6)$. As we can see, the actual trajectory of the robot coincides with the reference trajectory. Fig. 4.16a shows the linear and angular velocities of the mobile robot (which are the outputs of the controller); they remain within accepted limits[1]). Fig. 4.16b represents the projection of the four points in the image plane, where their initial locations are marked with (o), and their final locations are marked with (+).

Remark 9 *In the previous simulations, we assumed that the camera is perfectly calibrated, i.e. intrinsic parameters used in camera projection are identical to those used in pose estimation. Adding an error of about* 5% *to intrinsic parameters* α_x *and* α_y *will result in a 'drift' from the refernce trajectory as shown in Fig. 4.17. This problem is the main drawback of position-based visual servoing.*

[1] $|v|_{max} = 1$ and $|\omega|_{max} = 1$



(a) Trajectory **(b)** Velocities

Figure 4.18: Image-based visual servoing using fixed reference image features.

Image-based Visual Servoing

We start by using the same four points given above as target configuration. Initial robot posture is $(x_0, y_0, \theta_0) = (0, 1, 0)$, and desired image features are calculated as the projection of these points on the image plane when the robot is at the desired pose $(x_f, y_f, \theta_f) = (2.0m, 1.0m, 0.0rad)$. In this case the four points correspond to $(X_{d1}, Y_{d1}) = (0.3200, -0.1075)$,$(X_{d2}, Y_{d2}) = (-0.3200, -0.1075)$,$(X_{d3}, Y_{d3}) = (-0.3200, 0.7425)$ and $(X_{d4}, Y_{d4}) = (0.3200, 0.7425)$[1]. These values are used to estimate Image Jacobian (4.53) instead of the real-time values of image features.

Fig. 4.18 shows that the robot reaches its final destination and the controller outputs (v and ω) are within limits. Fig. 4.19 shows the evolution of the image features in the image plane: the four point locations on the image converge toward the desired locations marked by (x). This convergence, however, is not guaranteed for all initial postures, because it is possible that the system converges to a local minimum, as we can see for example in Fig. 4.20, where initial posture is chosen as $(x_0, y_0, \theta_0) = (0, 0, 0)$.

Now instead of the fixed reference we plan a reference trajectory of image features on the image plane as described in section 4.5. Initial posture is $(x_0, y_0, \theta_0) = (0, 0, 0)$ while final posture is $(x_f, y_f, \theta_f) = (3, 1, 0)$. Fig. 4.21a shows a perfect trajectory tracking

[1]Normalized coordinates are used as usual

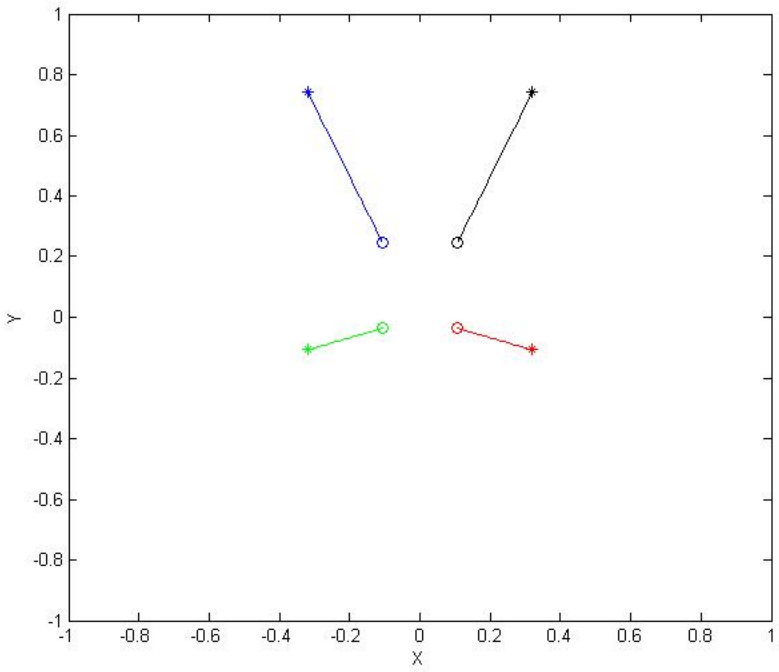


Figure 4.19: Points locations seen from the camera; initial positions (o), final positions (+).

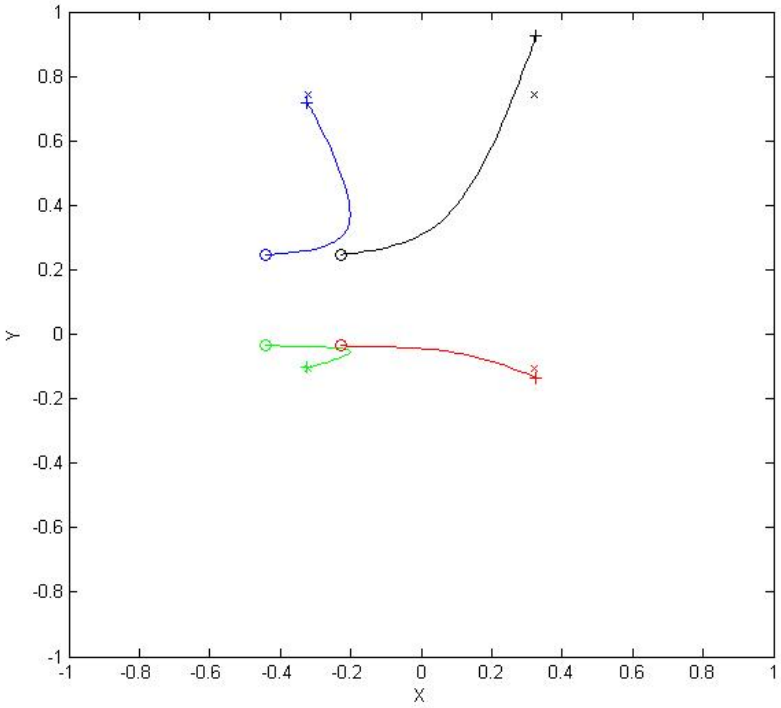
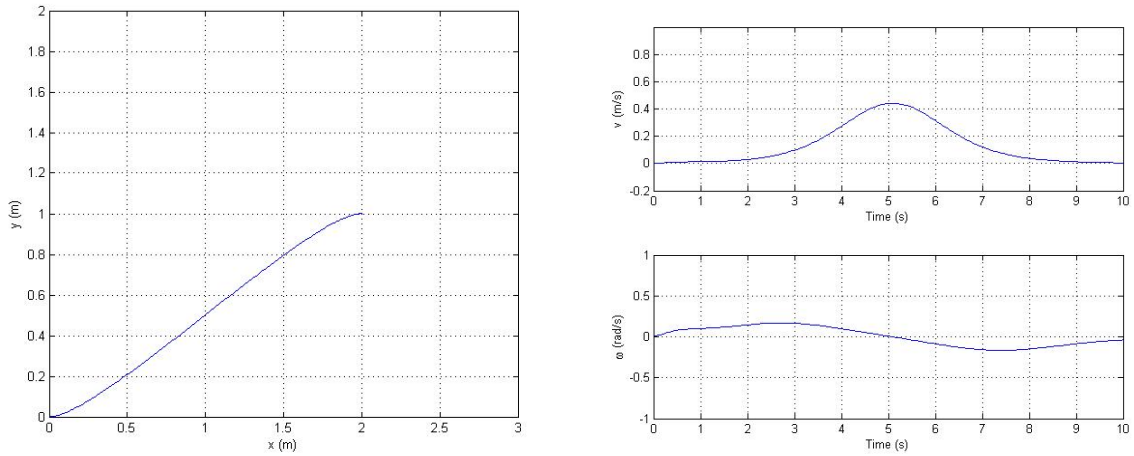


Figure 4.20: Points locations seen from the camera; initial positions (o), final positions (+) and desired positions (x).



(a) Trajectory **(b)** Velocities

Figure 4.21: Image-based visual servoing using planned reference trajectory in image plane.

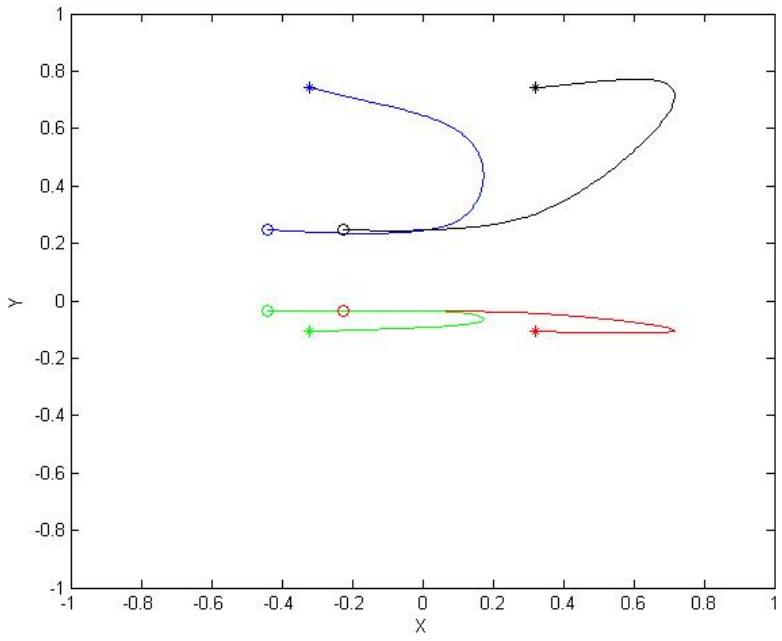


Figure 4.22: Points locations seen from the camera; initial positions (o), final positions (+) and desired positions (x).

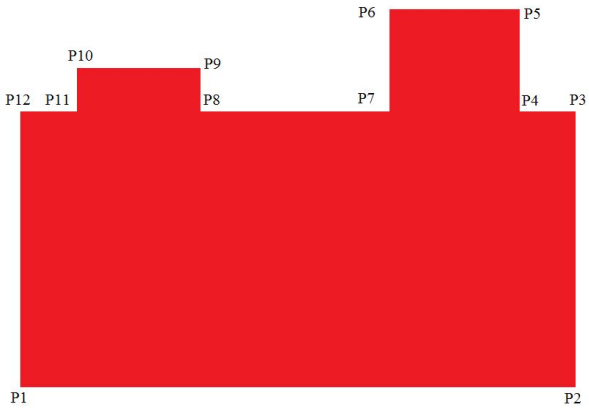


Figure 4.23: Target object used for visual servoing using image moments.

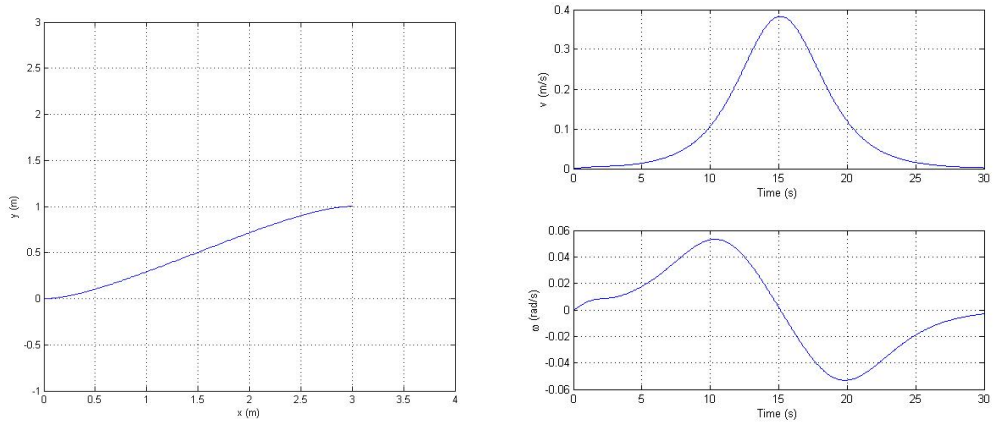
in the plane (x, y), while Fig. 4.22 shows trajectory tracking in the image plane. This result is obtained regardless of initial position provided that planned trajectories respect velocities limits and guarantee that the target object remains in the camera's field of view.

Now we will use basic image moments $m_{i,j}$ as image features, for this purpose let's consider the coplanar polygon shown in Fig. 4.23 as target object. We consider that this object is situated on a vertical plane parallel to (y, z) plane and corresponds to $x = 4$, its vertices' coordinates in the world frame are given in Table 4.1.

Point	(x_o, y_o, H_o)	Point	(x_o, y_o, H_o)	Point	(x_o, y_o, H_o)
p_1	(4.0, 0.9, 0.325)	p_2	(4.0, 1.1, 0.325)	p_3	(4.0, 1.1, 0.42)
p_4	(4.0, 1.075, 0.42)	p_5	(4.0, 1.075, 0.445)	p_6	(4.0, 1.03, 0.445)
p_7	(4.0, 1.03, 0.42)	p_8	(4.0, 0.965, 0.42)	p_9	(4.0, 0.965, 0.465)
p_{10}	(4.0, 0.92, 0.465)	p_{11}	(4.0, 0.92, 0.42)	p_{12}	(4.0, 0.9, 0.42)

Table 4.1: Target object vertices in world frame.

We plan a trajectory in the image plane using equations (4.75), this trajectory corresponds to a Cartesian trajectory planned via cubic polynomials. Initial posture is chosen as $(x_0, y_0, \theta_0) = (0, 0, 0)$, while final posture is $(x_f, y_f, \theta_f) = (3, 1, 0)$. Fig. 4.24a shows the trajectory in the (x, y) plane, while Fig. 4.25 shows the evolution of image moment values which follow almost perfectly their reference values.



(a) Trajectory **(b)** Velocities

Figure 4.24: Image-based visual servoing using image moments.

4.7 Conclusion

In this chapter we have addressed visual servoing techniques, where two main approaches are introduced, namely position-based visual servoing and image-based visual servoing. Position-based visual servoing is sensitive to camera calibration errors. In fact the presence of uncertainties on calibration parameters, both intrinsic and extrinsic, produces errors on the estimate of operational space variables that may be seen as an external disturbance acting on the feedback path of the control loop, where disturbance rejection capability is low. Moreover, in PBVS the object geometry must be known because it is necessary for pose estimation.

On the other hand, in Image-based visual servoing the quantities used for the computation of the control action are directly defined in the image plane and measured in pixel units. This implies that the uncertainty affecting calibration parameters can be seen as a disturbance acting on the forward path of the control loop, where disturbance rejection capability is high. IBVS does not require, in principle, knowledge of the object geometry. However, the knowledge of object geometry and perfect camera calibration becomes very important when planning a trajectory in the image plane, which requires the calculation of the projection of object points on the image. This can be mended by the calculation of these projections through a learning phase, where the robot is

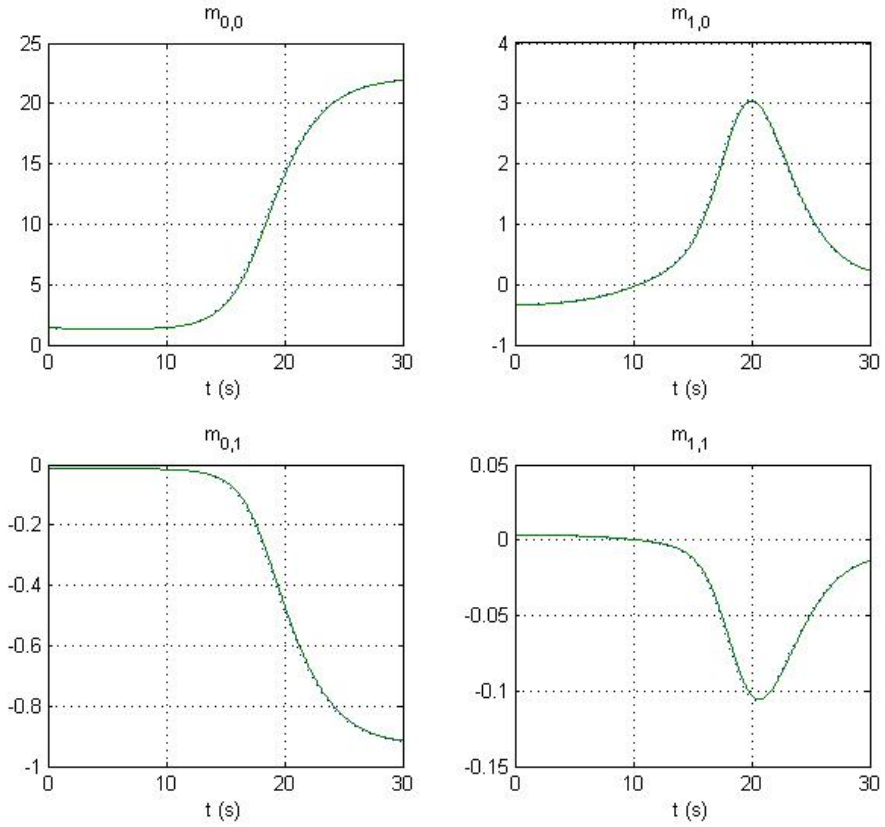


Figure 4.25: Evolution of the image moments of the robot Pekee II using Image-based visual servoing.

moved along the desired trajectory while measuring and storing the image features values (points or image moments). Nevertheless, this new approach eliminates the need for a learning step and target object movement can be easily taken into account.

Chapter 5

Experimental Results

The experimental validation tests have been first performed on the mobile robot Koala, the ability to apply the command directly on its wheels' motors makes it ideal for control strategy that takes into account a dynamical model, like the Integral sliding mode control strategy discussed in Section 3.3. However, its fixed camera and bad image quality makes it not very suitable for testing visual servoing techniques. In order to validate our results of visual servoing we resorted to use the wheeled mobile robot Pekee II. In what follows we will briefly present the robots' characteristics and the experiments that have been performed on each of them.

5.1 Wheeled Mobile Robot Koala

Koala is a mid-size wheeled mobile robot, it rides on 6 wheels for indoor operations. It has two differentially driven motorized wheels (the middle wheel of each side), with maximum speed 0.4 m/s, the wheels have a radius of 45mm and are mounted on an axle 30 cm long. The chassis of the robot measures 30x30x20 cm (l/w/h) and its total weight is 3.6 kg (4kg with battery). Each motor is equipped with an incremental encoder counting 5850 pulses/turn. The robot is equipped with 16 Infra-red proximity and ambient light sensors in addition to a camera mounted on a turret. Koala is provided for control purposes with a Motorola 68331@22MHz processor.

In order to control the robot, commands can be sent from a remote PC via serial port (RS232), and the robot responds either by performing an action (like turning the wheels for example), or by sending back data (e.g. encoders readings). DC motors can be

controlled directly by adjusting the duty cycle of the applied PWM signal of each motor, or indirectly via embedded PID position and speed controllers. Image acquisition from the camera is done via a graphic card (Blackboard) connected to the PC. Control and visual servoing algorithms are implemented using C++ language. Image processing and segmentation are performed using a special library called MIRAGE developed by Supelec[1]. The complete control architecture schematic of the robot Koala is shown in Fig. 5.1.

5.1.1 Experiments

In this section, we will report the experimental results of Koala in tracking an eight shape reference trajectory defined by

$$\begin{aligned} x_d(t) &= x_{dmax}\sin(2\pi\frac{t}{T}) + x_{d0} \\ y_d(t) &= y_{dmax}\sin(2\pi\frac{t}{2T}) + y_{d0} \end{aligned} \tag{5.1}$$

for $t \in [0, 2T]$. We choose $x_{dmax} = 2m$, $y_{dmax} = 2m$, $(x_{d0}, y_{d0}) = (1, 0)$ and $T = 40s$. From which we obtain using (2.10)

$$\begin{aligned} \theta_d(0) &= 0.4636rad \\ v_d(0) &= 0.3512m/s \\ \omega_d(0) &= 0rad/s \\ v_{dmax} &= 0.3512m/s \\ \omega_{dmax} &= 0.4580rad/s \end{aligned}$$

We apply feedback linearization discussed in section 3.2.1, we let $q(0) = (0m, 0m, 0rad)$, i.e., starting with an initial state error with respect to the assigned trajectory $q_d(0) = (1.0m, 0.0m, 0.4636rad)$. Data acquisition and control implementation are performed at a sampling period $T_s = 0.05s$.

In the first set of experiments, a PID controller is applied with $k_p = 9.17$,$k_i = 0.72$ and $k_d = 10.59$. As we can see from Figs. 5.2 and 5.3, relatively high tracking errors (up to 10.0 cm) are observed on x, y and $0.5rad$ on θ. These tracking errors are resulting from unmodeled dynamics (motors dynamics and unmodeled friction forces) and measurements errors. In addition, there is a large transient error resulting from the initial posture being different from the desired trajectory starting point.

[1]Metz Campus

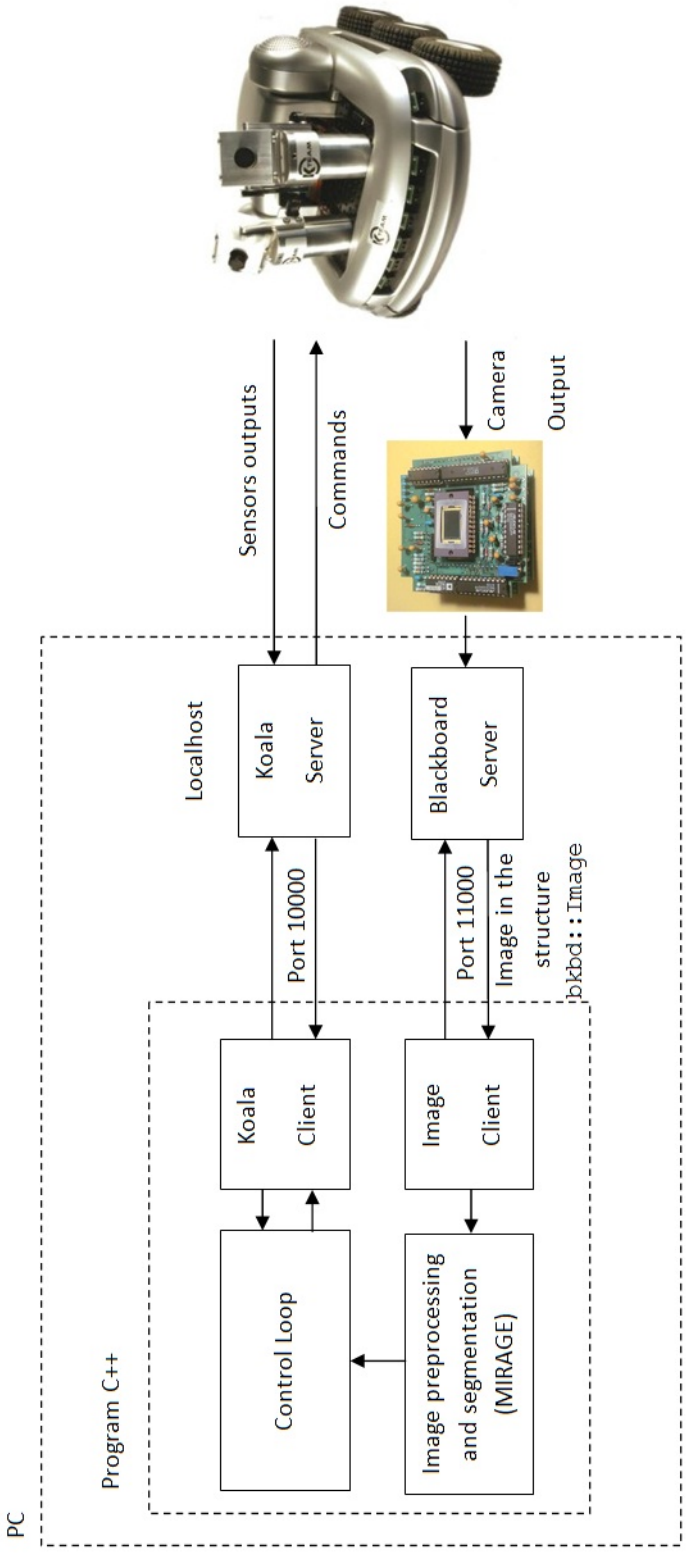


Figure 5.1: System architecture schematic of the robot Koala.

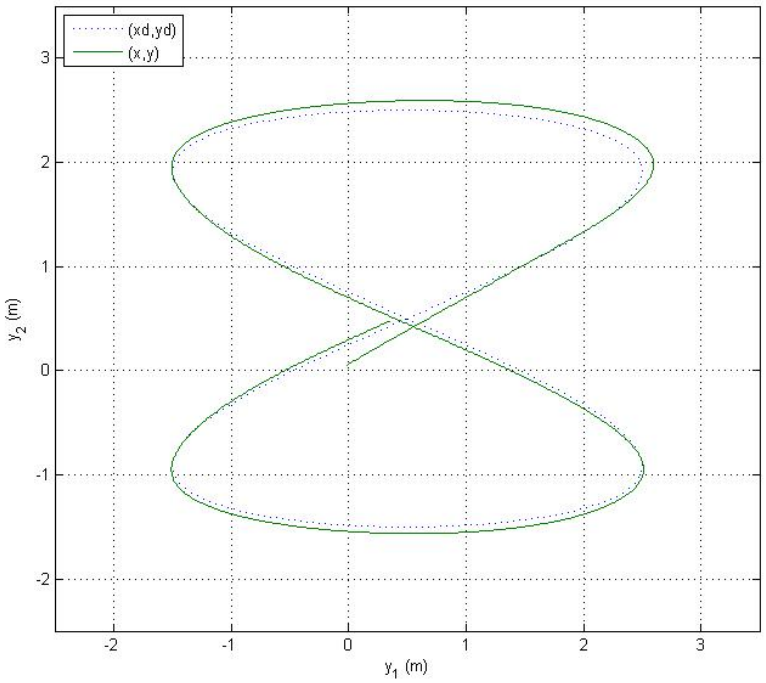
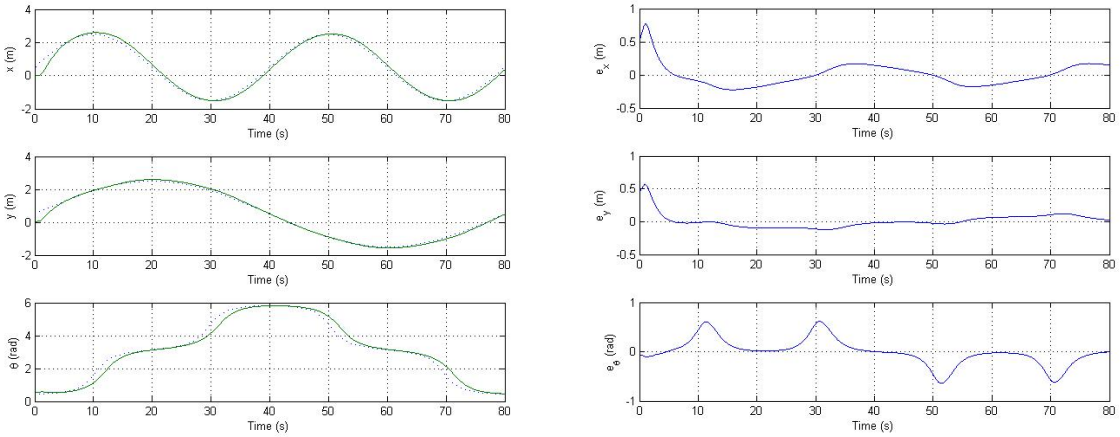
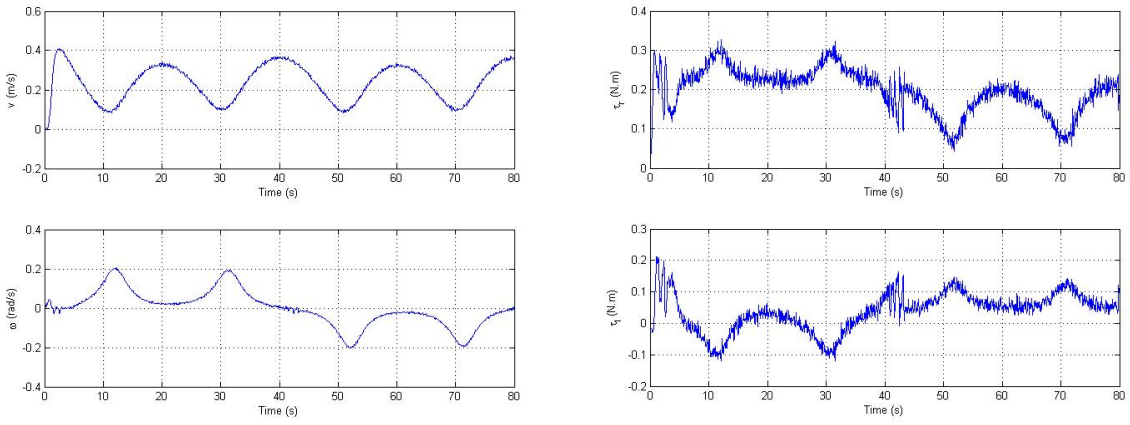


Figure 5.2: Asymptotic trajectory tracking using linear PID controller to track an eight shape trajectory on the (x, y) plane.



(a) Trajectory on x,y and θ **(b)** Tracking Errors

Figure 5.3: Asymptotic trajectory tracking using PID controller.



(a) Linear and angular velocities **(b)** Applied torques

Figure 5.4: Asymptotic trajectory tracking using PID controller.

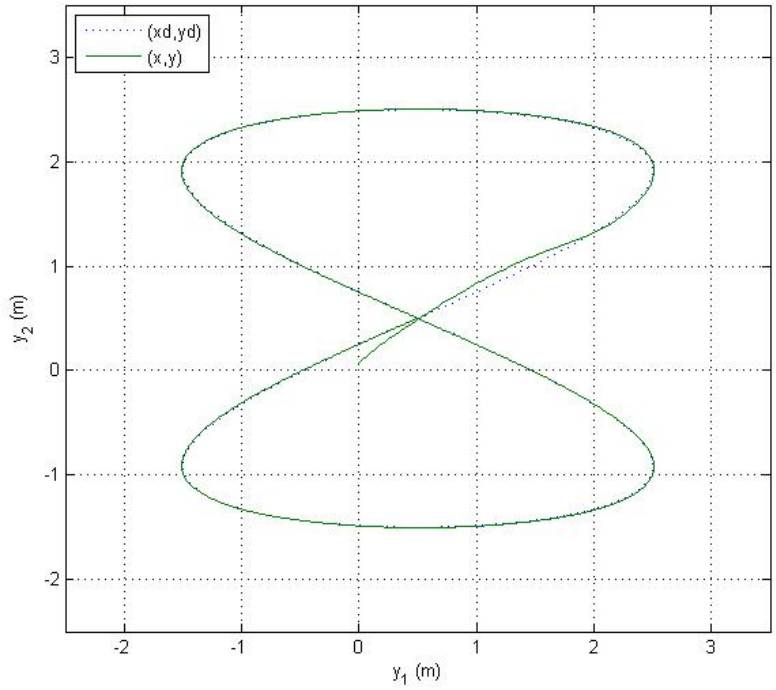
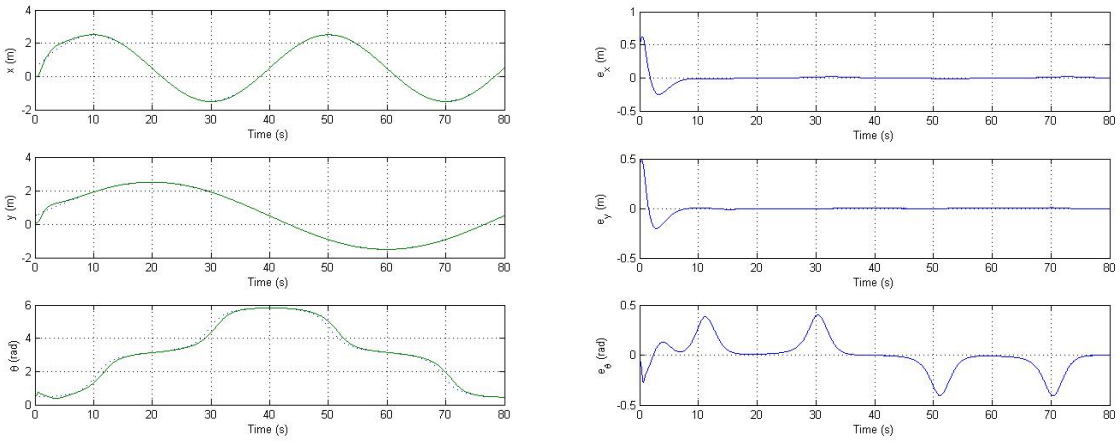


Figure 5.5: Asymptotic trajectory tracking using Integral Sliding Mode controller of an eight shape trajectory on the (x, y) plane.

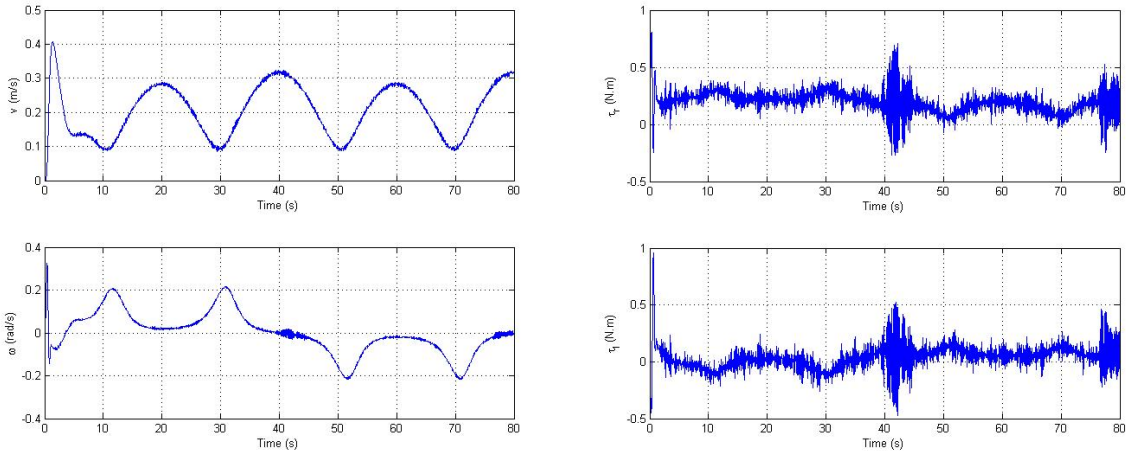
In the second set of experiments, we add Integral Sliding Mode controller as described in Section 3.3.3. As we can see from Figs. 5.5 and 5.6, the tracking of the reference trajectory is quite accurate. Residual errors (1 cm Maximum) are mainly due to quantization and discretization of velocity commands.

Another experiment was conducted to track trajectories generated using cubic polynomials between two points in presence of obstacles. The modified **TangentBug** algorithm described in Section 2.3 is used to avoid obstacles that are detected using Infra-red sensors. Fig. 5.8 shows the implementation of this algorithm in presence of a cylindrical object on the path between the start point and the target point. As we can see, a smooth path is generated when the obstacle is detected.



(a) Trajectory on x,y and θ **(b)** Tracking Errors

Figure 5.6: Asymptotic trajectory tracking using Integral Sliding Mode controller.



(a) Linear and angular velocities **(b)** Applied torques

Figure 5.7: Asymptotic trajectory tracking using Integral Sliding Mode controller.

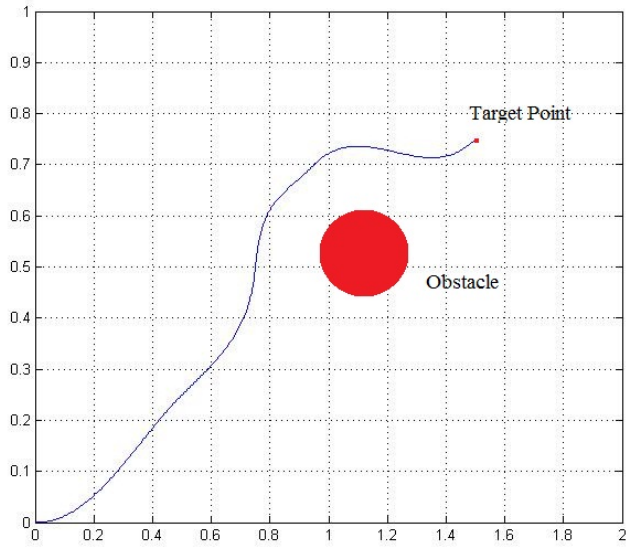


Figure 5.8: Trajectory tracking in presence of obstacles.

5.2 Wheeled Mobile Robot Pekee II

Pekee II from Wany Robotics[1], is a mid-size, cylinder-shaped wheeled mobile robot designed for indoor applications. It has two differentially driven wheels whose diameter is 72mm placed on the diameter of the robot. The cylindrical chassis of the robot has a diameter of 387mm, height 176mm (245mm with the camera) and its total weight is approximately 8.0 kg. The two 12V DC motors of Pekee II are equipped with a gear box, allowing to get a maximum speed of 250mm/s with the maximum torque of 3.0Nm. The positions of the wheels are measured using two odometers (255560 tops per wheel cycle). The robot is equipped with 8 ultra sonic sensors in addition to a pan-tilt Axis 214 PTZ camera which provides a video output with a resolution up 704x576 with a frame rate up to 30fps. Pekee II is provided for control purposes with an embedded PC that runs under Microsoft Windows XP.

In order to control the robot, commands can be sent from a remote PC via WiFi 802.11 connection, and the robot responds either by performing an action (e.g. moving forward or turning around), or by sending back data (e.g. odometers' readings or

[1]http://www.wanyrobotics.com/.

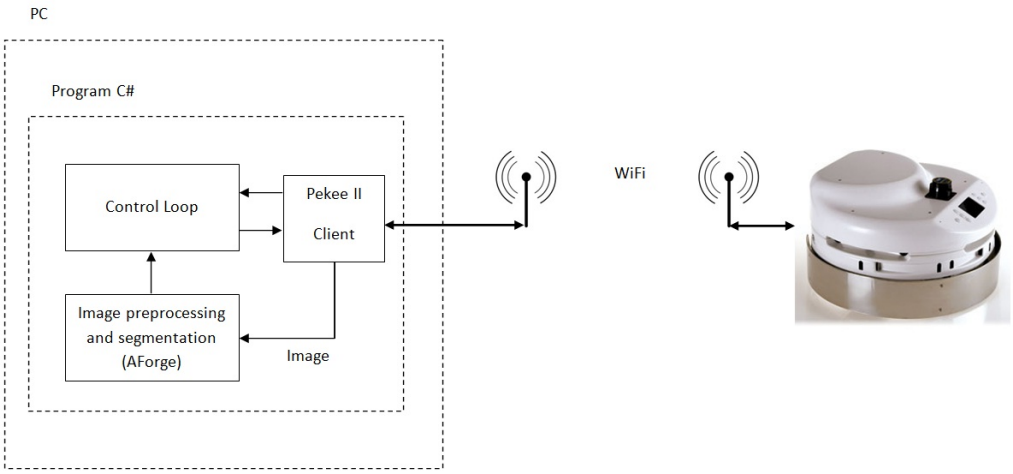


Figure 5.9: System architecture schematic of the robot Pekee II.

camera streaming). Communications between the robot and the remote PC is achieved by a standard TCP/IP protocol, this protocol allows for a server/multiple clients platform, which makes multiple robots coordination possible in the future. Control and visual servoing algorithms are implemented using C# language. Image processing and segmentation are performed using AForge[1] framework. Fig. 5.9 shows the complete control architecture schematic of the robot Pekee II.

5.2.1 Camera Calibration

The method of Zhang presented in appendix C was used to calibrate the camera with an image resolution of 640×480. The pattern used is a plane on which was printed a checkboard pattern of 8 by 8 squares. The items selected are the corners of each square. Each image of the plane provides for 49 points. Sixteen images used are shown in Figure 5.10. They show the model under different directions and at different distances from the optical center as illustrated in Fig. 5.11. The aim is to calculate a set of parameters for points distributed in a fairly large area around the camera as this will be the case for objects observed in an indoor environment.

[1]C# framework designed for developers and researchers in the fields of Computer Vision and Artificial Intelligence, downloadable from http://code.google.com/p/aforge/.

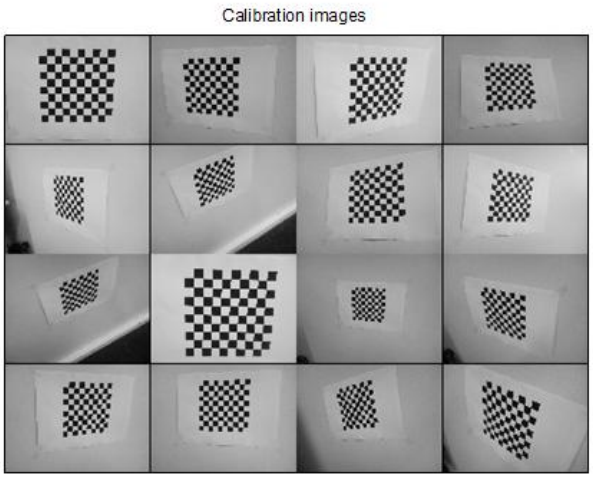


Figure 5.10: Calibration pattern taken from different directions and at different distances from the camera.

Intrinsic parameter values of robot Pekee II's camera obtained by calibration are listed in Table 5.1.

Parameter	Value	Uncertainty
α_x	747.18	$\pm$20.52(2.75%)
α_y	749.17	$\pm$18.47(2.47%)
u_0	311.32	$\pm$10.5(3.37%)
v_0	234.54	$\pm$8.55(3.65%)
δ	90.0	$\pm$0.0(0.0%)

Table 5.1: Intrinsic parameter values of robot Pekee II's camera obtained by calibration.

5.2.2 Experiments

Experiments have been conducted on the wheeled mobile robot Pekee II in order to validate theoretical and simulation results of chapters 3 and 4.

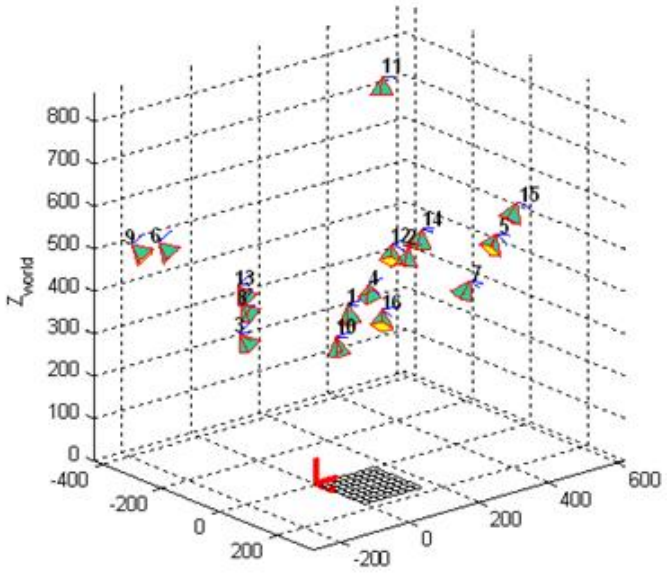


Figure 5.11: Camera's position and orientation relative to the pattern.

A. Posture Stabilization using I&I controller

In this case, the desired posture is the origin $(x_d, y_d, \theta_d) = (0, 0, 0)$. At $t = 0$ sec, the initial position of the robot is $(x(0), y(0), \theta(0)) = (-1.98, -3.0, \pi/2)$. We implemented the controller (3.67) and the control gains are tuned in order to achieve fast convergence toward the desired manifold and to satisfy limitations on linear and angular velocities. In our simulations they are set to $k = 0.8$, $\lambda_1 = 1.2$ and $\lambda_2 = -4.3$.

Fig. 5.12 shows the robot's trajectory in the (x, y) plane, while Fig.5.13 shows robot's orientation and linear and angular velocities. As we can observe the controller ensures the stabilization of the Pekee II, and the robot reaches its desired posture with practically no static errors.

B. Trajectory tracking using I&I controller

The tracking performance of the I&I control strategy is verified in the tracking of the reference trajectory that consists of successive straight line segments and arc line segments which may represent the output of a typical path planner. Reference and actual trajectories of the robot are depicted in Fig. 5.14, while tracking errors are

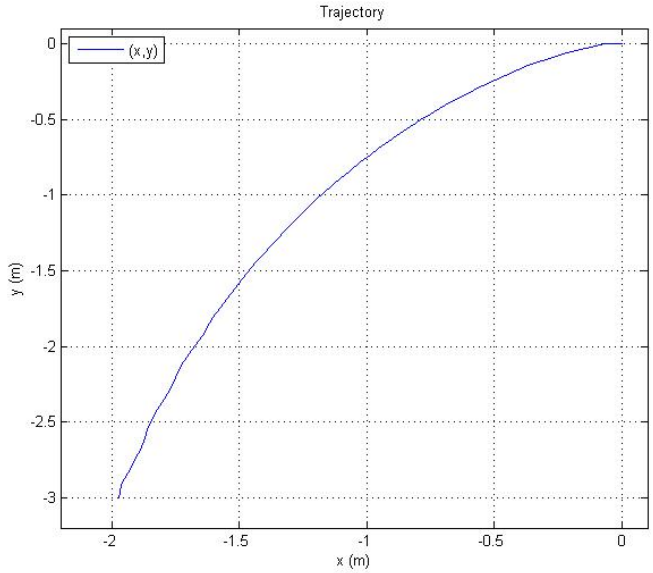
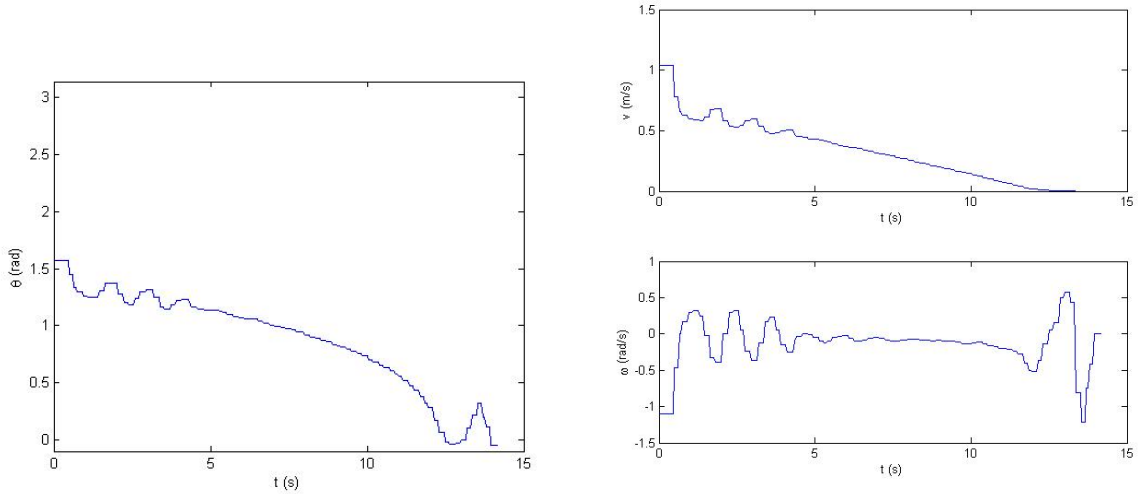


Figure 5.12: Stabilization of Pekee II using $I\&I$ controller for $(x_0, y_0, \theta_0) = (-1.98, -3.0, \pi/2)$.



(a) Orientation **(b)** Linear and angular velocities

Figure 5.13: Stabilization of Pekee II using $I\&I$ controller for $(x_0, y_0, \theta_0) = (-1.98, -3.0, \pi/2)$.

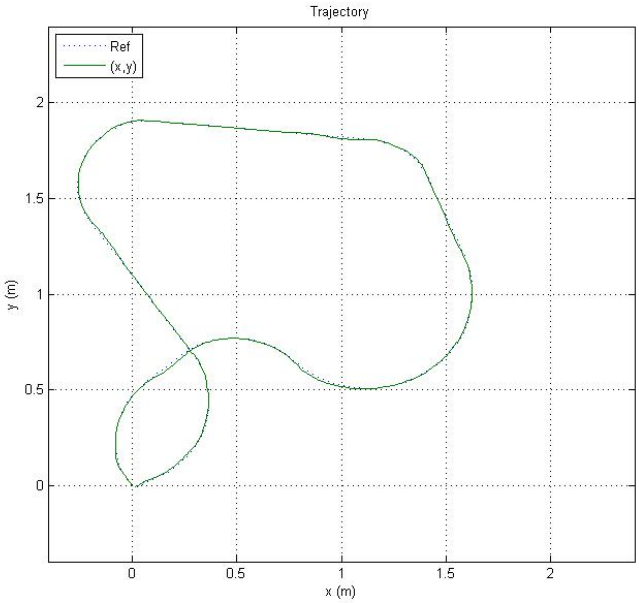


Figure 5.14: Trajectory tracking of robot Pekee II using I&I controller.

shown in Fig. 5.15a. Clearly the mobile robot is able to track the reference trajectory. Residual errors are mainly due to quantization and discretization of velocity commands.

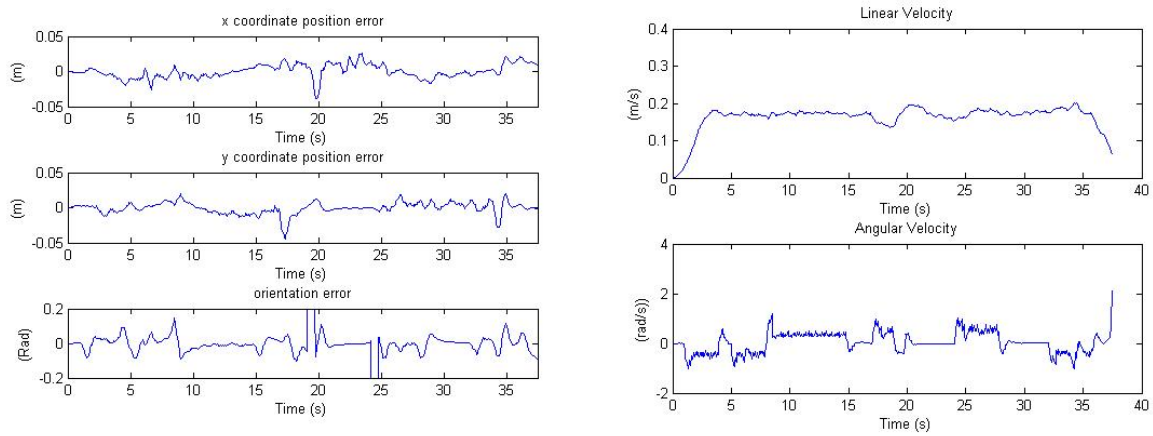
C. Position-based visual servoing

The target under consideration consists of four colored marks on the vertices of a stationary planar square (Fig. 5.16). This object is situated on a vertical plane parallel to (y, z) plane and corresponds to $x = 3m$, its vertice coordinates in world frame are given in Table 5.2.

Point	Color	Coordinates in world frame
p_1	Red	(3.0m,-0.08m,0.15m)
p_2	Green	(3.0m,0.08m,0.15m)
p_3	Blue	(3.0m,0.08m,0.20m)
p_4	Black	(3.0m,-0.08m,0.20m)

Table 5.2: Target object vertices in world frame.

A standard computer vision segmentation algorithm extracts the marks from the



(a) Tracking errors **(b)** Linear and angular velocities

Figure 5.15: Trajectory tracking of robot Pekee II using I&I controller.

background and computes the central moment of each mark. The central moments are transformed into unit-norm spherical image plane representation using the camera calibration matrix. The initial posture is chosen as $(x_0, y_0, \theta_0) = (0, -0.4m, 0)$, and the desired posture is $(x_0, y_0, \theta_0) = (2.4m, 0, 0)$. The trajectory is then generated using cubic polynomials.

Fig. 5.17 shows the robot's trajectory in the Cartesian space, while Fig. 5.18 shows the evolution of the image features in the image plane. By observing the tracking errors in both Cartesian space and image plane shown in Fig. 5.19 we note that the

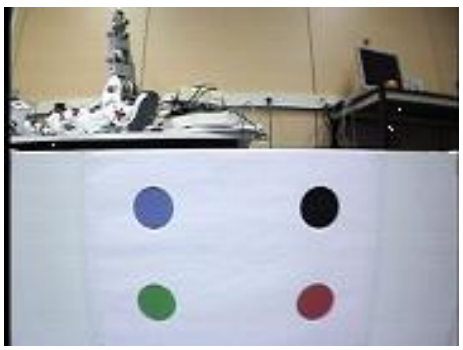


Figure 5.16: Target pattern used in Position-based visual servoing.

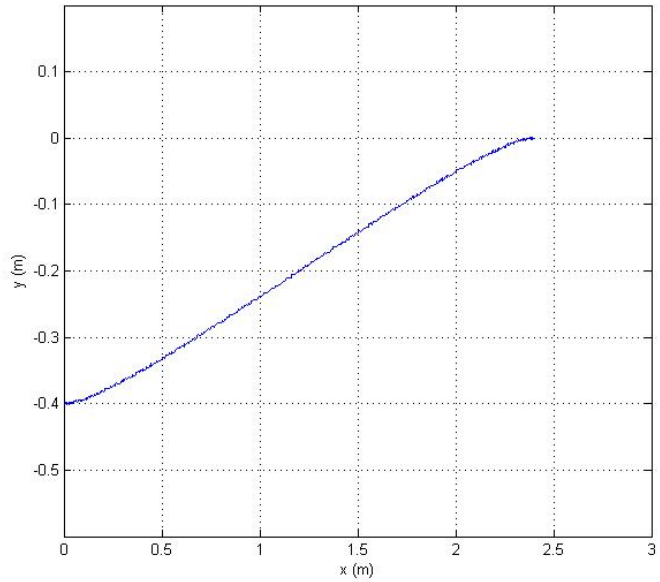


Figure 5.17: Trajectory on the plane (x, y) of the robot Pekee II using Position-based visual servoing.

position-based visual servoing control strategy ensures a good trajectory tracking, as long as the target object (all four circles) remains in the camera field of view, which is ensured by carefully choosing reference trajectory parameters.

Remark 10 *It is worth noting that x and y values shown in Fig. 5.17 were not obtained using odometers readings, but by pose estimation using image features shown in Fig. 5.18. In reality there exists a small difference between both as a direct result of camera calibration uncertainties and image noise.*

D. Image-based visual servoing

In the experiments we used the same pattern (Fig. 4.23) used in simulations with its vertices coordinates in world frame given in Table 5.3.

We plan a trajectory in the image plane using equations (4.75), this trajectory corresponds to a Cartesian trajectory planned via cubic polynomials. Initial posture is chosen as $(x_0, y_0, \theta_0) = (0, -0.80, 0)$, while final posture is $(x_f, y_f, \theta_f) = (2.5, 0, 0)$.

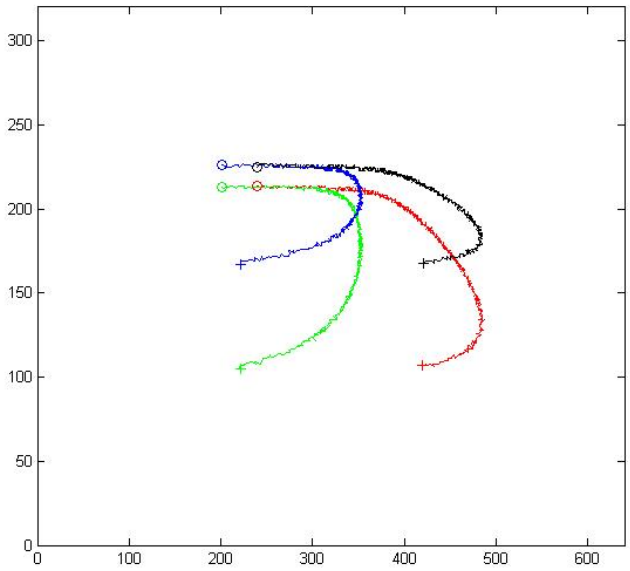
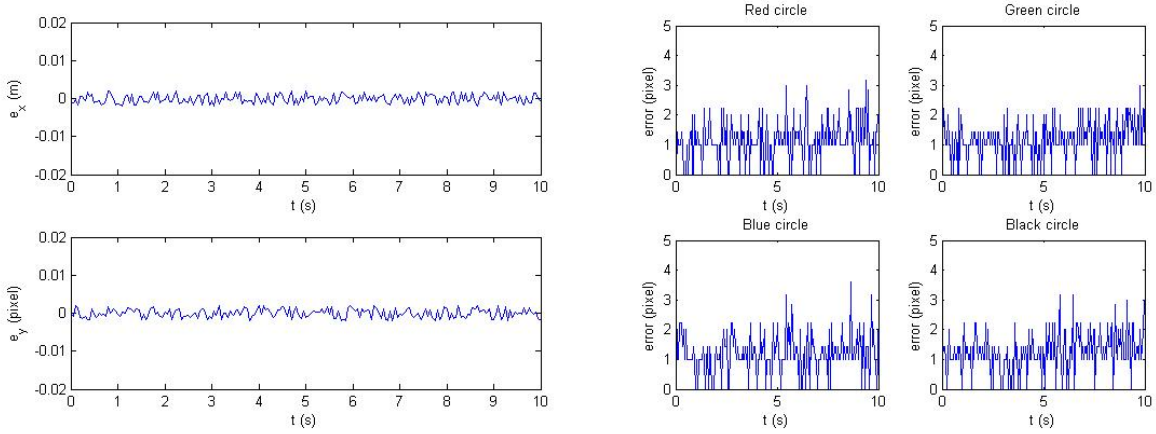


Figure 5.18: Evolution of the image features in pixels in the image plane of the robot Pekee II using Position-based visual servoing.



(a) Tracking errors on (x, y)

(b) Tracking errors in the image

Figure 5.19: Trajectory errors of the robot Pekee II using Position-based visual servoing.

Point	(x_o, y_o, H_o)	Point	(x_o, y_o, H_o)	Point	(x_o, y_o, H_o)
p_1	(3.0, 0.1, 0.325)	p_2	(3.0, -0.1, 0.325)	p_3	(3.0, -0.1, 0.42)
p_4	(3.0, -0.075, 0.42)	p_5	(3.0, -0.075, 0.445)	p_6	(3.0, -0.03, 0.445)
p_7	(3.0, -0.03, 0.42)	p_8	(3.0, 0.35, 0.42)	p_9	(3.0, 0.35, 0.465)
p_{10}	(3.0, 0.12, 0.465)	p_{11}	(3.0, 0.12, 0.42)	p_{12}	(3.0, 0.3, 0.42)

Table 5.3: Target object vertices in world frame.

These values of image moments are then stored within memory and used in the control loop as shown in Fig. 4.12 in Section 4.5.

Fig. 5.20 shows the trajectory in the (x, y) plane, while Figs. 5.22 and 5.23 show the evolution of image moment values and their reference values. We note that our strategy in Image-based visual sevoing ensures a good trajectory tracking throughout the whole trajectory.

5.3 Conclusion

In this chapter, experiments have been conducted on two wheeled mobile robots: Koala and Pekee II. An Integral Sliding Mode based control strategy combined with state feedback linearization has been implemented on Koala, this controller provided far better results than a traditional PID controller in trajectory tracking and disturbances rejection. A reactive path planning algorithm was implemented to take advantage of the Infra-red sensors in order to detect static obstacles and provide a smooth collision-free path. A control strategy based on the application of *Immersion and Invariance* methodology has been implemented on the mobile robot Pekee II to address both stabilization and trajectory tracking. Satisfactory results have been obtained for both control problems. Position-based visual servoing strategy has been implemented, this control strategy ensured a good trajectory tracking, as long as the target object remains in the camera field of view, which is guaranteed by carefully choosing reference trajectory parameters, the difference between the trajectory calculated using pose estimation and the one calculated using odometers is due to camera calibration uncertainties. Image-based visual servoing strategy is then implemented using image moments as visual features, this visual servoing strategy is combined with trajectory tracking on

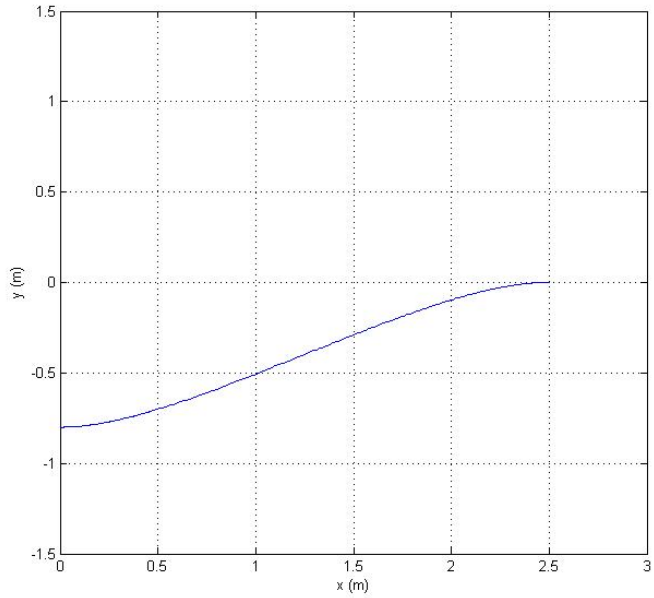
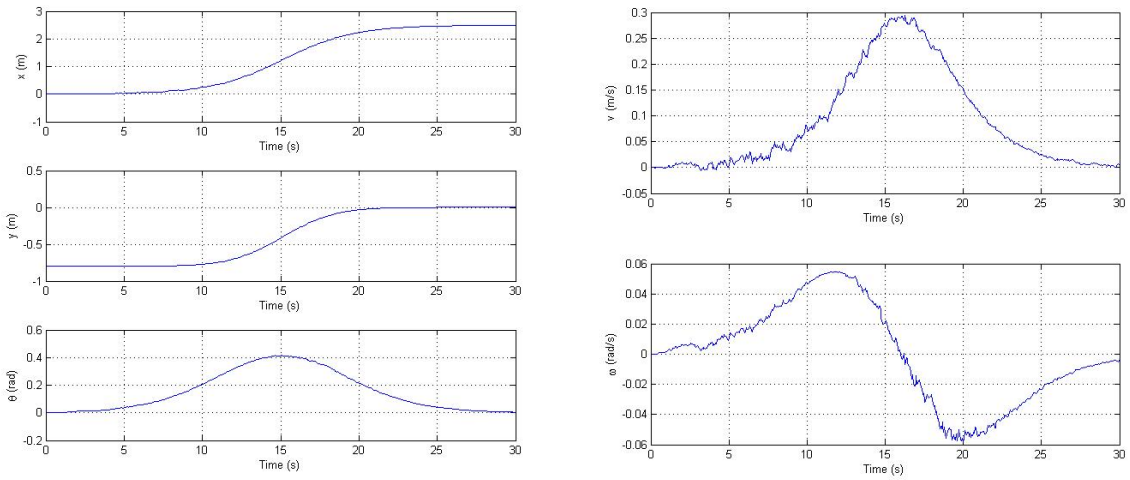
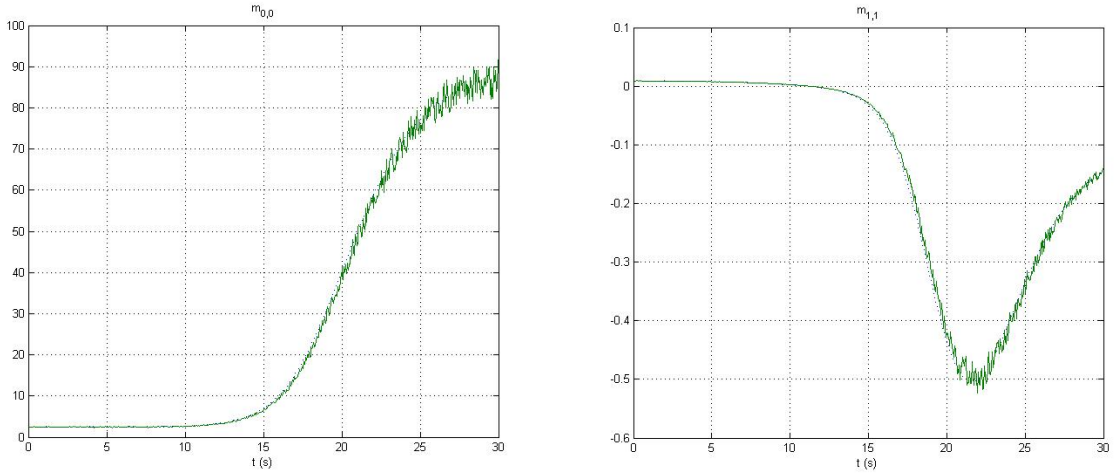


Figure 5.20: Trajectory on the plane (x, y) of the robot Pekee II using Image-based visual servoing.



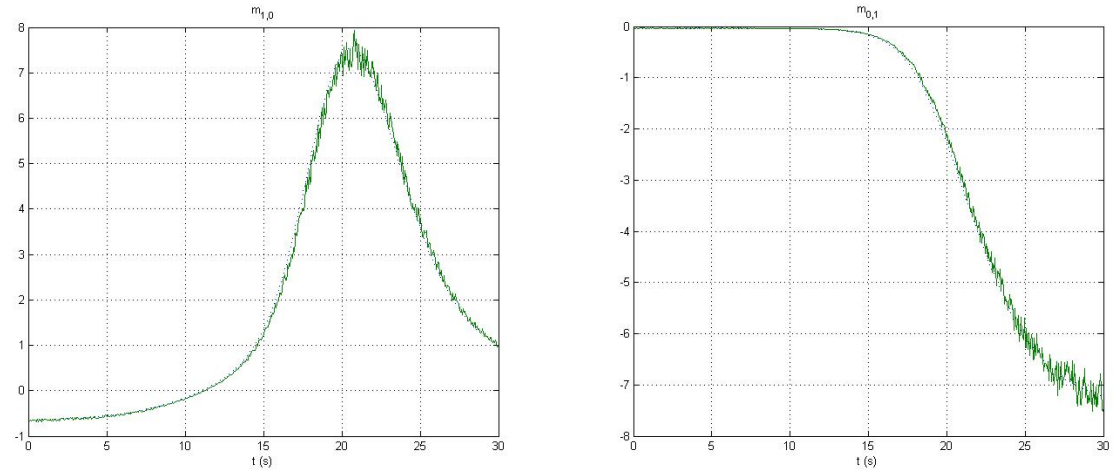
(a) Trajectory on x,y and θ **(b)** Linear and angular velocities

Figure 5.21: Evolution of the image moments of the robot Pekee II using Image-based visual servoing.



(a) m00 **(b)** m11

Figure 5.22: Evolution of the image moments of the robot Pekee II using Image-based visual servoing.



(a) m10 **(b)** m01

Figure 5.23: Evolution of the image moments of the robot Pekee II using Image-based visual servoing.

the image plane, experiments showed a good trajectory tracking throughout the whole trajectory.

Conclusions and Perspectives

In this work, we discussed the motion control problem and visual servoing of a nonholonomic wheeled mobile robot. First, we presented kinematic and dynamic models of the robot in addition to their control properties. We discussed then the trajectory planning problem in the presence of nonholonomic constraints. The existence of flat outputs, is then exploited to implement trajectory planning methods that guarantee that the nonholonomic constraints are satisfied. Path planning in the presence of static obstacles is then discussed, a modified *TangentBug* algorithm was implemented in order to generate smooth collision-free trajectories.

Next, we discussed the motion control problem for a nonholonomic mobile robot, with reference to two basic motion tasks, i.e., posture regulation and trajectory tracking. Two control approaches have been described: First, state feedback Linearization is extended to include both kinematic and dynamic models, or the so called second-order kinematic model. A control strategy is applied, on the basis of the Integral Sliding Mode approach with the objective of posture regulation and tracking pre-generated trajectories. This combination has led to satisfactory results in terms of stabilization and robustness. Second, a control strategy based on the application of *Immersion and Invariance* methodology is described. This method was used to derive a class of controllers for both control problems. The proposed control strategy guarantees that the closed-loop system asymptotically behaves like a given target system achieving asymptotic model matching, which makes the system performance adjustment simpler and physically meaningful. Stability of the system has been guaranteed by appropriate choice of the target systems. Simulations have been conducted which established the quality of both control strategies.

We then addressed visual servoing techniques, where two main approaches are introduced, namely position-based visual servoing and image-based visual servoing. The

use of the so-called image moments as image feature parameters in both approaches is also presented. A new approach to image-based visual servoing is developed, which is based on the trajectory generation in the image plane directly (computing image features values corresponding to a given Cartesian trajectory by computing the projection in the image of a 3D model of the target for the desired camera pose). This approach guarantees that the robustness and stability of image-based servoing have been extended due to the fact that initial and desired camera locations are close to each other. The obtained trajectories ensure that the target remains in the camera field of view and that the corresponding robot motion is physically realizable. This method doesn't require a learning step, which has the drawback to depend on position estimation using encoders informations (which may be unavailable or unreliable), and to it requires that the target object remains fixed throughout this phase. In our method however, target object movement can be easily incorporated into our equations.

Finally, we conducted experiments to validate our results of motion control and visual servoing techniques on two mobile robots: Koala and Pekee II. Technical aspects of both implementations have been briefly presented. Satisfactory results have been obtained from both implementations regarding motion control and visual servoing strategies.

Perspectives

Future work should explore other possibilities and routes in which this thesis's work could head to. Main future work additions could be:

- **Implementation of other control algorithms**: We have implemented two control strategies for both navigation problems, one based on the Integral Sliding mode, and the other on the Immersion and Invariance. Adaptive control strategies could be investigated for example, in order to solve the stabilization and tracking control of nonholonomic systems in presence of parametric and nonparametric uncertainties.

- **Further experimental testing considering more complex environments**: Experiments were conducted but within limited sized and simple environments, always considering static objects. Future testing should include larger and more

complex environments, possibly with inclusion of moving objects. Also, experimental testing should be incorporated earlier in the development process.

- **Identification of more than one target object per captured image**: The target object recognition using computer vision was implemented with the restriction on the number of objects that can be identified in a captured image. Future work should consider the identification of multiple target objects within a captured image.

- **Considering dynamic environments**: In path planning, we have considered static environments whose contents remain fixed for the whole duration of navigation. Future path planning should consider dynamic environments. The development of autonomous exploration strategies, should provide solutions to the problems of sustainable change in the environment. Regarding the changes to the environment, it is useful to treat them as a problem of obstacle avoidance. In this approach, it would probably be interesting to perform an estimation of the speed of the obstacles.

- **Considering the problem of coordinating multiple robots**: We have considered in our implementation a single robot. The WiFi functionality of robot Pekee II gives the possibility of controlling multiple robots using a coordinating program running on the same remote computer.

Appendix A

Nonholonomic Constraints

Wheels are by far the most common mechanism to achieve locomotion in mobile robots. Any wheeled vehicle is subject to kinematic constraints that reduce in general its local mobility, while leaving intact the possibility of reaching arbitrary configurations by appropriate manoeuvers. For example, any driver knows by experience that, while it is impossible to move instantaneously a car in the direction orthogonal to its heading, it is still possible to park it arbitrarily, at least in the absence of obstacles. It is therefore important to analyze in details the structure of these constraints.

Let's consider a mechanical system whose configuration $\mathbf{q} \in \mathcal{C}$ is described by a vector of generalized coordinates, and assume that the configuration space $\mathcal{C}$ (i.e., the space of all possible robot configurations) coincides with $\mathbb{R}^n$. The motion of the system that is represented by the evolution of $\mathbf{q}$ over time may be subject to constraints that can be classified under various criteria. For example, they may be expressed as equalities or inequalities (respectively, bilateral or unilateral constraints), and they may depend explicitly on time or not (rheonomic or scleronomic constraints). Only bilateral scleronomic constraints will be considered. Constraints that can be put in the form

$$h_i(\mathbf{q}) = 0 \qquad i = 1, 2 \cdots k < n \tag{A.1}$$

are called *holonomic* (or integrable), where functions $h_i : \mathcal{C} \longmapsto \mathbb{R}$ are of class C^∞(smooth) and independent. The effect of holonomic constraints is to reduce the space of accessible configurations to a subset of $\mathcal{C}$ with dimension $n - k$. A mechanical system for which all the constraints can be expressed in form (A.1) is called *holonomic*.

Constraints that involve generalized coordinates and velocities

$$a_i(\mathbf{q}, \dot{\mathbf{q}}) = 0 \qquad i = 1, 2 \cdots k < n \tag{A.2}$$

are called *kinematic*. They constrain the instantaneous admissible motion of the mechanical system by reducing the set of generalized velocities that can be attained at each configuration. Kinematic constraints are generally expressed in Pfaffian form, i.e., they are linear in the generalized velocities:

$$\mathbf{a}_i^T(\mathbf{q})\dot{\mathbf{q}} = 0 \qquad i = 1, 2 \cdots k < n \tag{A.3}$$

or, in matrix form

$$\mathbf{A}^T(\mathbf{q})\dot{\mathbf{q}} = 0 \tag{A.4}$$

Clearly, the existence of k holonomic constraints (A.1) implies an equal number of kinematic constraints:

$$\frac{dh_i}{dt} = \frac{\partial h_i}{\partial \mathbf{q}}\dot{\mathbf{q}} = 0 \qquad i = 1, 2 \cdots k < n \tag{A.5}$$

However, the inverse is not true in general. A system of kinematic constraints in form (A.3) may or may not be integrable to form (A.1). In the negative case, the kinematic constraints are said to be *nonholonomic* (or nonintegrable). A mechanical system that is subject to at least one such constraint is called *nonholonomic*.

A.1 Integrability Conditions

In the presence of Pfaffian kinematic constraints, integrability conditions can be used to decide whether the system is holonomic or nonholonomic.

Consider first the case of a *single* Pfaffian constraint:

$$\mathbf{a}_i^T(\mathbf{q})\dot{\mathbf{q}} = \sum_{j=1}^{n} a_j(\mathbf{q})\dot{q}_j \tag{A.6}$$

For this constraint to be integrable, there must exist a scalar function $h(\mathbf{q})$ and an integrating factor $\gamma(\mathbf{q}) \neq 0$ such that the following condition holds:

$$\gamma(\mathbf{q})a_j(\mathbf{q}) = \frac{\partial h_(\mathbf{q})}{\partial \mathbf{q}_j} \qquad j = 1, 2 \cdots n \tag{A.7}$$

The converse is also true: if there exists an integrating factor $\gamma(\mathbf{q}) \neq 0$ such that $\gamma(\mathbf{q})\mathbf{a}(\mathbf{q})$ is the gradient of a scalar function $h(\mathbf{q})$, constraint (A.6) is integrable. By using Schwarz theorem on the symmetry of second derivatives, integrability condition (A.7) may be replaced by the following system of partial differential equations:

$$\frac{\partial \gamma \mathbf{a}_k}{\partial \mathbf{q}_j} = \frac{\partial \gamma \mathbf{a}_j}{\partial \mathbf{q}_k} \qquad j,k = 1,2\cdots n,\ j \neq 0 \tag{A.8}$$

Condition (A.8) implies that a Pfaffian constraint with constant coefficients a_j is always holonomic.

A.2 Brockett's Theorem

Theorem 2 *Let $x = f(x,u)$ be given with $f(x_0,0) = 0$ and $f(.,.)$ continuously differentiable in a neighborhood of $(x,0)$. A necessary condition for the existence of a continuously differentiable control law which makes $(x_0,0)$ asymptotically stable is that:*

1. *the linearized system should have no uncontrollable modes associated with eigenvalues whose real part is positive.*

2. *there exists a neighborhood N of $(x_o,0)$ such that for each $\xi \in N$ there exists a control $u_\xi(.)$ defined on $[0,\infty)$ such that this control steers the solution of $\dot{x} = f(x,u_\xi)$ from $x = \xi$ at $t = 0$ to $x = x_0$ at $t \to \infty$.*

3. *the mapping $\gamma : A \times \mathbb{R}^m \to \mathbb{R}^n$ defined by*

$$\gamma : (x,u) \longmapsto f(x,u)$$

should be onto an open set containing 0.

Proof 2 *(see [17])*

If the control system is of the form

$$\dot{x} = \sum_1^m g_i(x)u_i, \qquad x(t) \in N \subset \mathbb{R}^n$$

with vectors $g_i(x)$ being linearly independent at x_0 then condition (3) implies that there exists a solution to the stabilization problem *if and only if* $m = n$. In this case we must have as many control parameters as we have dimensions of x.

A.3 Definition of function atan2

atan2 is a two argument function defined as follows

$$\text{atan2}(y,x) = \begin{cases} \arctan(\frac{y}{x}) & x > 0 \\ \pi + \arctan(\frac{y}{x}) & y \geq 0, x < 0 \\ -\pi + \arctan(\frac{y}{x}) & y < 0, x < 0 \\ \frac{\pi}{2} & y > 0, x = 0 \\ -\frac{\pi}{2} & y < 0, x = 0 \\ undefined & y = 0, x = 0 \end{cases} \tag{A.9}$$

Appendix B

Image Moments: Properties and Calculation

B.1 Green's Theorem

Let $\mathcal{C}$ be a positively oriented, piecewise smooth, simple closed curve in the plane $\mathbb{R}^2$, and let $\mathcal{R}$ be the region bounded by $\mathcal{C}$. If N and M are functions of (x, y) defined on an open region containing $\mathcal{D}$ and have continuous partial derivatives there, then [97]

$$\oint_{\mathcal{C}} (N\,\mathrm{d}x + M\,\mathrm{d}y) = \iint_{\mathcal{R}} \left(\frac{\partial M}{\partial x} - \frac{\partial N}{\partial y} \right) \mathrm{d}x\,\mathrm{d}y \tag{B.1}$$

Considering only two-dimensional vector fields, Green's theorem is equivalent to the following two-dimensional version of the divergence theorem

$$\iint_{\mathcal{R}} (\nabla \cdot \mathbf{F})\, dA = \oint_{C} \mathbf{F} \cdot \hat{\mathbf{n}}\, ds, \tag{B.2}$$

where $\hat{\mathbf{n}}$ is the outward-pointing unit normal vector on the boundary.

To see this, consider the unit normal in the right side of the equation. Since $d\mathbf{r} = (dx, dy)$ is a vector pointing tangential along a curve, and the curve $\mathcal{C}$ is the positively-oriented (i.e. counterclockwise) curve along the boundary, an outward normal would be a vector which points 90° to the right, which would be $(dy, -dx)$. The length of this vector is $\sqrt{dx^2 + dy^2} = ds$. So $\hat{\mathbf{n}}\, ds = (dy, -dx)$.

Now let the components of $\mathbf{F} = (P, Q)$. Then the right hand side becomes

$$\int_{\mathcal{C}} \mathbf{F} \cdot \hat{\mathbf{n}}\, ds = \int_{\mathcal{C}} (P dy - Q dx) \tag{B.3}$$

which by Green's theorem becomes

$$\int_{\mathcal{C}}(-Qdx + Pdy) = \iint_D \left(\frac{\partial P}{\partial x} + \frac{\partial Q}{\partial y}\right) dA = \iint_D (\nabla \cdot \mathbf{F})\, dA \tag{B.4}$$

B.2 Moments of two dimensional functions

For a 2-D continuous function $f(x, y)$, the moment of order (i, j) is defined as [42]

$$m_{i,j} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x, y)x^i y^j dxdy \tag{B.5}$$

for $i, j = 0, 1, 2.....$ A uniqueness theorem (see [79]) states that if $f(x, y)$ is piecewise continuous and has nonzero values only in a finite part of the plane (x, y), moments of all orders exist, and moments $m_{i,j}$ are uniquely determined by $f(x, y)$, Conversely $m_{i,j}$ uniquely determine $f(x, y)$.

The *central moments* are defined as

$$\mu_{i,j} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x, y)(x - x_g)^i (y - y_g)^j dxdy \tag{B.6}$$

where

$$x_g = \frac{m_{1,0}}{m_{0,0}}, \qquad y_g = \frac{m_{0,1}}{m_{0,0}}$$

If $f(x, y)$ is a digital image, then equation (B.6) becomes

$$\mu_{i,j} = \sum_x \sum_y f(x, y)(x - x_g)^i (y - y_g)^j \tag{B.7}$$

By developing (B.7) we obtain the central moments of order up to 3 as functions of basic moments

$$\begin{aligned}
\mu_{0,0} &= m_{0,0} \\
\mu_{1,0} &= 0 \\
\mu_{0,1} &= 0 \\
\mu_{1,1} &= m_{1,1} - x_g m_{0,1} = m_{1,1} - y_g m_{1,0} \\
\mu_{2,0} &= m_{2,0} - x_g m_{1,0} \\
\mu_{0,2} &= m_{0,2} - x_g m_{0,1} \\
\mu_{2,1} &= m_{2,1} - 2x_g m_{1,1} - y_g m_{2,0} + 2x_g^2 m_{0,1} \\
\mu_{1,2} &= m_{1,2} - 2y_g m_{1,1} - x_g m_{0,2} + 2y_g^2 m_{1,0}
\end{aligned} \tag{B.8}$$

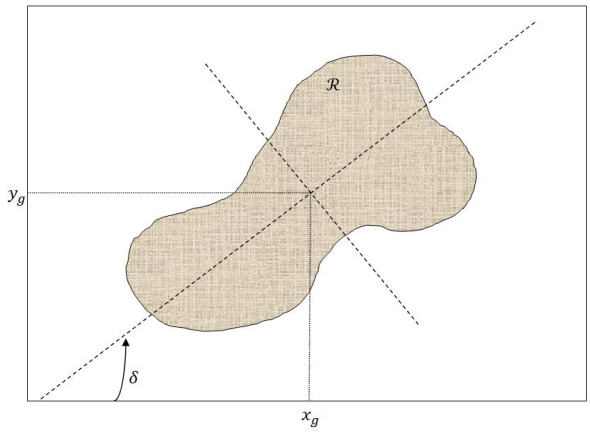


Figure B.1: Region of a binary image and some feature parameters.

In the case of binary images, by assuming the light intensity equal to one for all the points of region $\mathcal{R}$, and equal to zero for all the points not belonging to $\mathcal{R}$, the following simplified definition of moment is obtained

$$m_{i,j} = \sum_{X,Y \in \mathcal{R}} X^i Y^j \tag{B.9}$$

In this case, moments have physical meanings (Fig. B.1). For example $m_{0,0}$ represents the area of the region and (x_g, y_g) represents the centroid of the region. Using mechanical analogy, it is easy to recognize that the central moments of second order $\mu_{2,0}$ and $\mu_{0,2}$ have the meaning of inertia moments with respect to axes X and Y respectively, while $\mu_{1,1}$ is an inertia product, and the matrix

$$I = \begin{bmatrix} \mu_{2,0} & \mu_{1,1} \\ \mu_{1,1} & \mu_{0,2} \end{bmatrix}$$

has the meaning of inertia tensor relative to the center of mass [92]. The eigenvalues of matrix I define the principal moments of inertia, termed *principal moments* of the region and the corresponding eigenvectors define the principal axes of inertia, termed *principal axes* of the region. If region $\mathcal{R}$ is asymmetric, the principal moments of I are different and it is possible to characterize the orientation of $\mathcal{R}$ in terms of the angle δ between the principal axis corresponding to the maximum moment and axis X. This

angle can be computed with the equation

$$\delta = \frac{1}{2}\arctan\left(\frac{2\mu_{1,1}}{\mu_{2,0} - \mu_{0,2}}\right) \tag{B.10}$$

B.3 Image moments of a polygon

Let's consider a non-self-intersecting (simple) coplanar polygon $\mathcal{P}$ with k vertices (x_1, y_1), $(x_2, y_2) \cdots (x_k, y_k)$, and let's define polygon edges in terms of a parameter t that ranges from zero to one. The evolutions of x, y along the edge between the two vertices (x_i, y_i), (x_{i+1}, y_{i+1}) are given by

$$\begin{aligned} Ex &= x_i + (x_{i+1} - x_i)t \\ Ey &= y_i + (y_{i+1} - y_i)t \end{aligned} \tag{B.11}$$

The differentials dx and dy are then given by

$$\begin{aligned} dx &= (x_{i+1} - x_i)dt \\ dy &= (y_{i+1} - y_i)dt \end{aligned} \tag{B.12}$$

Let's consider the moment $m_{0,0}$ of the polygon which is given by

$$m_{0,0} = \iint_{\mathcal{P}} dxdy \tag{B.13}$$

Using Green's formula (B.1), and by choosing $M = -y/2$ and $N = x/2$ we obtain

$$\iint_{\mathcal{P}} dxdy = \oint_{\mathcal{C}} (-\frac{1}{2}ydx + \frac{1}{2}xdy) \tag{B.14}$$

Evaluating the line integral along the edge of the polygon from (x_i, y_i) to (x_{i+1}, y_{i+1}), we have

$$I_i = \int_0^1 (-Eydx + Exdy)dt \tag{B.15}$$

Substituting (B.11) and (B.12) in (B.15), and after simple development, we obtain

$$I_i = \frac{1}{2}(x_i y_{i+1} - x_{i+1} y_i) \tag{B.16}$$

And $m_{0,0}$ is then given by

$$m_{0,0} = \frac{1}{2}\sum_{i=1}^{k}(x_i y_{i+1} - x_{i+1} y_i) \tag{B.17}$$

with $(x_{k+1}, y_{k+1}) = (x_0, y_0)$.

Similarly, we can obtain higher order image moments

$$
\begin{aligned}
m_{1,0} =& \frac{1}{6}\sum_{i=1}^{k}\left(x_i + x_{i+1}\right)\left(x_i y_{i+1} - x_{i+1} y_i\right) \\
m_{0,1} =& \frac{1}{6}\sum_{i=1}^{k}\left(y_i + y_{i+1}\right)\left(x_i y_{i+1} - x_{i+1} y_i\right) \\
m_{2,0} =& \frac{1}{12}\sum_{i=1}^{k}(x_i^3 + x_i^2 x_{i+1} + x_i x_{i+1}^2 + x_{i+1}^3)(y_{i+1} - y_i) \\
m_{0,2} =& \frac{1}{12}\sum_{i=1}^{k}(y_i^3 + y_i^2 y_{i+1} + y_i y_{i+1}^2 + y_{i+1}^3)(x_{i+1} - x_i) \\
m_{1,1} =& \frac{1}{24}\sum_{i=1}^{k}\left[x_i^2 y_{i+1}\left(2y_i + y_{i+1}\right) - x_{i+1}^2 y_i\left(y_i + 2y_{i+1}\right) + 2x_i x_{i+1}\left(y_{i+1}^2 - y_i^2\right)\right] \\
m_{2,1} =& \frac{1}{60}\sum_{i=1}^{k}(y_{i+1} - y_i)(4x_i^3 y_i + x_i^3 y_{i+1} + x_{i+1}^3 y_i + 4x_{i+1}^3 y_{i+1} \\
& + 2x_i x_{i+1}^2 y_i + 3x_i^2 x_{i+1} y_i + 3x_i x_{i+1}^2 y_{i+1} + 2x_i^2 x_{i+1} y_{i+1}) \\
m_{1,2} =& \frac{1}{60}\sum_{i=1}^{k}(x_{i+1} - x_i)(4x_i y_i^3 + x_{i+1} y_i^3 + x_i y_{i+1}^3 + 4x_{i+1} y_{i+1}^3 \\
& + 2x_i y_i y_{i+1}^2 + 3x_i y_i^2 y_{i+1} + 3x_{i+1} y_i y_{i+1}^2 + 2x_{i+1} y_i^2 y_{i+1})
\end{aligned}
\tag{B.18}
$$

B.4 Image moments of an ellipse

In that case image projection is also an ellipse $\mathcal{E}$ whose antipodal points on its major axis and minor axis are denoted (X_a, Y_a), (X_{-a}, Y_{-a}), (X_b, Y_b) and (X_{-b}, Y_{-b}). Its elements are given as functions of the coordinates of these four points as follows

$$
\begin{aligned}
2a &= \sqrt{(X_a - X_{-a})^2 + (Y_a - Y_{-a})^2} \\
2b &= \sqrt{(X_b - X_{-b})^2 + (Y_b - Y_{-b})^2} \\
x_g &= \frac{1}{2}(X_a + X_{-a}) \\
y_g &= \frac{1}{2}(Y_a + Y_{-a}) \\
\delta &= \operatorname{atan2}(Y_a - Y_{-a}, X_a - X_{-a})
\end{aligned}
\tag{B.19}
$$

where $2a$, $2b$ are the major and minor diameters of the ellipse respectively, (x_g, y_g) is its center and δ is the angle between the major axis of the ellipse and the x-axis.

The evolutions of x, y along the ellipse are given by

$$\begin{aligned} Ex &= x_g + a\cos t\cos\delta - b\sin t\sin\delta \\ Ey &= y_g + a\cos t\sin\delta + b\sin t\cos\delta \end{aligned} \tag{B.20}$$

where t is a parametric variable that varies between 0 and 2π. The differentials dx and dy are then given by

$$\begin{aligned} dx &= (-a\sin t\cos\delta - b\cos t\sin\delta)dt \\ dy &= (-a\sin t\sin\delta + b\cos t\cos\delta)dt \end{aligned} \tag{B.21}$$

Let's consider the moment $m_{0,0}$ of the ellipse which is given by

$$m_{0,0} = \iint_{\mathcal{E}} dxdy \tag{B.22}$$

Using Green's formula (B.1), and by choosing $M = x$ we obtain

$$\begin{aligned} \iint_{\mathcal{E}} dxdy &= \oint_{\mathcal{C}} xdy \\ &= \int_0^{2\pi} (a\cos t\cos\delta - b\sin t\sin\delta)(-a\sin t\sin\delta + b\cos t\cos\delta)dt \\ &= \pi ab \end{aligned} \tag{B.23}$$

Similarly, we can obtain higher order image moments of an ellipse

$$\begin{aligned} m_{1,0} &= \pi abx_g \\ m_{0,1} &= \pi aby_g \\ m_{1,1} &= \frac{\pi}{8}ab\left(8x_gy_g + \left(a^2 - b^2\right)\sin 2\delta\right) \\ m_{2,0} &= \frac{\pi}{8}ab\left(8x_g^2 + a^2 + b^2 + \left(a^2 - b^2\right)\cos 2\delta]\right) \\ m_{0,2} &= \frac{\pi}{8}ab\left(8y_g^2 + a^2 + b^2 - \left(a^2 - b^2\right)\cos 2\delta]\right) \\ m_{2,1} &= \frac{\pi}{8}ab\left(\left(8x_g^2 + a^2 + b^2\right)y_g + \left(a^2 - b^2\right)\left(y_g\cos 2\delta + 2x_g\sin 2\delta\right)\right) \\ m_{1,2} &= \frac{\pi}{8}ab\left(\left(8y_g^2 + a^2 + b^2\right)x_g - \left(a^2 - b^2\right)\left(x_g\cos 2\delta - 2y_g\sin 2\delta\right)\right) \end{aligned} \tag{B.24}$$

Appendix C

Camera Calibration

For the calibration of the camera, we adopted an algorithm based on Zhang's method [110]. This technique only requires the camera to observe a planar pattern shown at a few (at least two) different orientations. In the camera's model used we consider the intrinsic parameters and the extrinsic parameters.

This method proposes an analytic solution to solve these parameters and then create the best approximation using a minimization method. An example of calibration plane with a chessboard pattern is shown in Fig. C.1.

The relationship between a 3D point $\tilde{\mathbf{p}}_b = [x \;\; y \;\; z \;\; 1]_b^T$ in base frame and its image projection $\tilde{s}_I = [u_I \;\; v_I \;\; 1]^T$ in pixels is given by

$$\lambda \tilde{\mathbf{s}}_I = \mathbf{K}[\mathbf{R}_b^c \;\; \mathbf{t}_b^c]\tilde{\mathbf{p}}_b \tag{C.1}$$

Without loss of generality, we assume the model plane is on $z = 0$ of the base coordinate system. From (C.1), we have

$$\begin{aligned} \lambda \begin{bmatrix} u_I \\ v_I \\ 1 \end{bmatrix} &= \mathbf{K} \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 & t_b^c \end{bmatrix} \begin{bmatrix} x \\ y \\ 0 \\ 1 \end{bmatrix} \\ &= \mathbf{K} \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & t_b^c \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \end{aligned} \tag{C.2}$$

By abuse of notation, we still use $\tilde{\mathbf{p}}_b$ to denote a point on the model plane, but $\tilde{\mathbf{p}} = [x \;\; y \;\; 1]^T$ since z is always equal to 0. Therefore, a model point $\tilde{\mathbf{p}}_b$ and its image

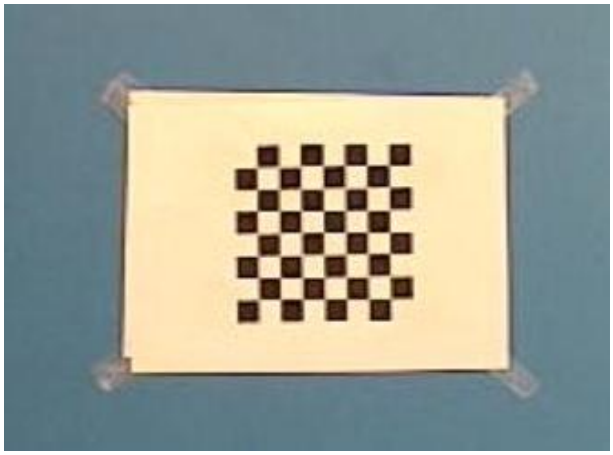


Figure C.1: Example of calibration plane.

$\tilde{\mathbf{s}}_I$ are related by a homography $\mathbf{H}$:

$$\lambda\tilde{\mathbf{s}}_I = \mathbf{H}\tilde{\mathbf{p}}_b \tag{C.3}$$

where $\mathbf{H} = \mathbf{K}[\mathbf{r}_1 \;\; \mathbf{r}_2 \;\; \mathbf{t}_b^c]$ is a 3×3 matrix defined up to a scale factor. Given an image of the model plane, an homography can be estimated (see [110]). Let's denote it by $\mathbf{H} = [\mathbf{h}_1 \;\; \mathbf{h}_2 \;\; \mathbf{h}_3]$. From (C.3), we have

$$[\mathbf{h}_1 \;\; \mathbf{h}_2 \;\; \mathbf{h}_3] = \rho\mathbf{K}[\mathbf{r}_1 \;\; \mathbf{r}_2 \;\; \mathbf{t}_b^c] \tag{C.4}$$

where ρ is an arbitrary scalar. Using the knowledge that $\mathbf{r}_1$ and $\mathbf{r}_2$ are orthonormal, we have

$$\mathbf{h}_1^T\mathbf{K}^{-T}\mathbf{K}^{-1}\mathbf{h}_2 = 0 \tag{C.5}$$

$$\mathbf{h}_1^T\mathbf{K}^{-T}\mathbf{K}^{-1}\mathbf{h}_1 = \mathbf{h}_2^T\mathbf{K}^{-T}\mathbf{K}^{-1}\mathbf{h}_2 \tag{C.6}$$

where we used the abbreviation $\mathbf{K}^{-T} = (\mathbf{K}^{-1})^T$. These are the two basic constraints on the intrinsic parameters, given one homography. Because a homography has 8 degrees of freedom and there are 6 extrinsic parameters (3 for rotation and 3 for translation), we can only obtain 2 constraints on the intrinsic parameters.

Now let's define $\mathbf{B}$

$$\begin{aligned}\mathbf{B} = \mathbf{K}^{-T}\mathbf{K}^{-1} &\equiv \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{12} & b_{22} & b_{23} \\ b_{13} & b_{23} & b_{33} \end{bmatrix} \\ &= \begin{bmatrix} \frac{1}{\alpha^2} & -\frac{\gamma}{\alpha^2\beta} & \frac{u_0\gamma - u_0\beta}{\alpha^2\beta} \\ -\frac{\gamma}{\alpha^2\beta} & \frac{\gamma^2}{\alpha^2\beta^2} + \frac{1}{\beta^2} & -\frac{\gamma(u_0\gamma - u_0\beta)}{\alpha^2\beta^2} - \frac{v_0}{\beta^2} \\ \frac{u_0\gamma - u_0\beta}{\alpha^2\beta} & -\frac{\gamma(u_0\gamma - u_0\beta)}{\alpha^2\beta^2} - \frac{v_0}{\beta^2} & \frac{(u_0\gamma - u_0\beta)^2}{\alpha^2\beta^2} + \frac{v_0}{\beta^2} + 1 \end{bmatrix}\end{aligned} \tag{C.7}$$

Since $\mathbf{B}$ is symmetric, we can define the 6D vector

$$\mathbf{b} = \begin{bmatrix} b_{11} & b_{12} & b_{22} & b_{13} & b_{23} & b_{33} \end{bmatrix}^T \tag{C.8}$$

If we denote $\mathbf{h}_i = [h_{i1} \;\; h_{i2} \;\; h_{i3}]$ the i^{th} colomn of matrix $\mathbf{H}$ then we have

$$\mathbf{h}_i^T \mathbf{B} \mathbf{h}_j = \mathbf{v}_{ij}^T \mathbf{b} \tag{C.9}$$

with

$$\mathbf{v}_{ij} = \begin{bmatrix} h_{i1}h_{j1} \\ h_{i1}h_{j2} + h_{i2}h_{j1} \\ h_{i2}h_{j2} \\ h_{i3}h_{j1} + h_{i1}h_{j3} \\ h_{i3}h_{j2} + h_{i2}h_{j3} \\ h_{i3}h_{j3} \end{bmatrix} \tag{C.10}$$

Therefore, the two fundamental constraints (C.5) and (C.6) from a given homography, can be rewritten as two homogeneous equations in $\mathbf{b}$

$$\begin{bmatrix} \mathbf{v}_{12}^T \\ (\mathbf{v}_{11} - \mathbf{v}_{22})^T \end{bmatrix} \mathbf{b} = \mathbf{0} \tag{C.11}$$

If n images of the model plane are observed, by stacking n such equations as (C.11) we have

$$\mathbf{V}\mathbf{b} = \mathbf{0} \tag{C.12}$$

where $\mathbf{V}$ is a $2n \times 6$ matrix. If $n \geq 3$, we will have in general a unique solution $\mathbf{b}$ defined up to a scale factor. Once $\mathbf{b}$ is estimated, we can compute all camera intrinsic

matrix $\mathbf{K}$ using (C.7), or more specifically

$$\begin{aligned} v_0 &= (b_{12}b_{13} - b_{11}b_{23})/(b_{11}b_{22} - b_{12}^2) \\ \rho &= b_{33} - \left(b_{13}^2 + v_0(b_{12}b_{13} - b_{11}b_{23})\right)/b_{11} \\ \alpha &= \sqrt{\rho/b_{11}} \\ \beta &= \sqrt{\rho b_{11}/((b_{11}b_{22} - b_{12}^2)} \\ \gamma &= -b_{12}\alpha^2\beta/\rho \\ u_0 &= \gamma v_0/\beta - b_{13}\alpha^2/\rho \end{aligned} \tag{C.13}$$

Once camera matrix K is recovered, we can recover the camera motion using (C.4) as follows

$$\begin{aligned} \mathbf{r}_1 &= \frac{\mathbf{K}^{-1}\mathbf{h}_1}{\|\mathbf{K}^{-1}\mathbf{h}_1\|} \\ \mathbf{r}_2 &= \frac{\mathbf{K}^{-1}\mathbf{h}_2}{\|\mathbf{K}^{-1}\mathbf{h}_2\|} \\ \mathbf{r}_3 &= \mathbf{r}_1 \times \mathbf{r}_2 \\ \mathbf{t}_b^c &= \frac{1}{2}\left(\frac{1}{\|\mathbf{K}^{-1}\mathbf{h}_1\|} + \frac{1}{\|\mathbf{K}^{-1}\mathbf{h}_1\|}\right)\mathbf{K}^{-1}\mathbf{h}_3 \end{aligned} \tag{C.14}$$

Generally, because of noise in data, the so-computed matrix $R = [r1 \;\; r2 \;\; r3]$ does not in general satisfy the properties of a rotation matrix. This can be refined through maximum likelihood inference. We are given n images of a model plane and there are m points on the model plane. We assume that the image points are corrupted by independent and identically distributed noise. The maximum likelihood estimate can be obtained by minimizing the following functional

$$\sum_{i=1}^{n}\sum_{j=1}^{m} \|\mathbf{m}_{ij} - \widehat{m}(\mathbf{K}, \mathbf{R}_{bi}^c, \mathbf{t}_{bi}^c, M_j)\| \tag{C.15}$$

where $\widehat{m}(\mathbf{K}, \mathbf{R}_{bi}^c, \mathbf{t}_{bi}^c, \mathbf{M}_j)$ is the projection of point Mj in image i, according to equation (C.3). A rotation $\mathbf{R}_b^c$ is parameterized by a vector of 3 parameters, denoted by $\mathbf{r}$, which is parallel to the rotation axis and whose magnitude is equal to the rotation angle. $\mathbf{R}$ and r are related by the Rodrigues formula [37]. Minimizing (C.15) is a nonlinear minimization problem, which is solved with the Levenberg-Marquardt Algorithm (see [69]). It requires an initial guess of $\mathbf{K}; \{\mathbf{R}_{bi}^c; \mathbf{t}_{bi}^c | i = 1 \cdots n\}$ which can be obtained using (C.14).

Bibliography

[1] J.Á. Acosta, R. Ortega, A. Astolfi, and I. Sarras. A constructive solution for stabilization via immersion and invariance: The cart and pendulum system. *Automatica*, 44(9):2352–2357, 2008. 62

[2] D. Aeyels. Stabilization of a class of nonlinear systems by a smooth feedback control. *Systems & Control Letters*, 5(5):289–294, 1985. 60

[3] O. Amidi, T. Kanade, R. Miller, O. Amidi, T. Kanade, and R. Miller. Vision-based autonomous helicopter research at carnegie mellon robotics institute 1991-1997. 1998. 4

[4] A. Astolfi. Exponential stabilization of a car-like vehicle. In *Proceedings of IEEE International Conference on Robotics and Automation*. IEEE, 1995. 2, 65

[5] A. Astolfi. Discontinuous control of nonholonomic systems. *Systems & Control Letters*, 27(1):37–45, 1996. 2, 65

[6] A. Astolfi. Exponential stabilization of a wheeled mobile robot via discontinuous control. *Journal of Dynamic Systems, Measurement, and Control*, 121:121–126, 1999. 3

[7] A. Astolfi and R. Ortega. Immersion and invariance- A new tool for stabilization and adaptive control of nonlinear systems. *IEEE Transactions on Automatic Control*, 48(4):590–606, 2003. xxii, 60

[8] A. Astolfi, D. Karagiannis, and R. Ortega. *Nonlinear and adaptive control with applications*. Springer Verlag, 2008. 6, 39, 60, 61

[9] R.N. Banavar and V. Sankaranarayanan. *Switched finite time control of a class of underactuated systems*. Springer, 2006. 3

[10] J. Barraquand and J.C. Latombe. On nonholonomic mobile robots and optimal maneuvering. In *Proceedings of IEEE International Symposium on Intelligent Control*, pages 340–347. IEEE, 1989. 2

[11] J. Batista, H. Araujo, and A.T. de Almeida. Iterative multistep explicit camera calibration. *IEEE Transactions on Robotics and Automation*, 15(5):897–917, 1999. 86

[12] Z. Bien, W. Jang, and J. Park. Characterization and use of feature jacobian matrix for visual servoing. *Visual Serving: Real Time Control of Robot Manipulators Based on Visual Sensory Feedback*, page 317, 1993. 4

[13] A. Bloch and S. Drakunov. Stabilization of a nonholonomic system via sliding modes. In *Proceedings of the 33rd IEEE Conference on Decision and Control*, volume 3, pages 2961–2963, 1994. 45

[14] A. Bloch and S. Drakunov. Tracking in nonholonomic dynamic systems via sliding modes. In *Proceedings of the 34th IEEE Conference on Decision and Control*, volume 3, pages 2103–2106. IEEE, 1995. 45

[15] AM Bloch and NH McClamroch. Control of mechanical systems with classical nonholonomic constraints. In *Proceedings of the 28th IEEE Conference on Decision and Control*, pages 201–205. IEEE, 1989. 2

[16] A.M. Bloch, M. Reyhanoglu, and N.H. McClamroch. Control and stabilization of nonholonomic dynamic systems. *IEEE Transactions on Automatic Control*, 37 (11):1746–1757, 1992. 2

[17] R.W. Brockett. Asymptotic stability and feedback stabilization, 1983. 2, 39, 143

[18] G. Campion, B. d'Andrea Novel, and G. Bastin. Controllability and state Feedback stabilizability of non holonomic mechanical systems. *Advanced robot control*, pages 106–124, 1991. 13, 24

[19] G. Campion, B. d'Andrea Novel, and G. Bastin. Modelling and state feedback control of nonholonomic mechanical systems. In *Proceedings of the 30th IEEE Conference on Decision and Control*, volume 2, pages 1184–1189, 1991. 24, 40

[20] G. Campion, G. Bastin, and B. Dandrea-Novel. Structural properties and classification of kinematic and dynamic models of wheeled mobile robots. *IEEE Transactions on Robotics and Automation*, 12(1):47–62, 1996. 1, 13

[21] F. Chaumette. Potential problems of stability and convergence in image-based and position-based visual servoing. *The confluence of vision and control*, pages 66–78, 1998. 94

[22] F. Chaumette. A first step toward visual servoing using image moments. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 1, pages 378–383. IEEE, 2002. 4

[23] F. Chaumette. Image moments: a general and useful set of features for visual servoing. *Robotics, IEEE Transactions on*, 20(4):713–723, 2004. 4, 81, 98

[24] K.L. Chung. Computing horizontal/vertical convex shape's moments on reconfigurable meshes. *Pattern Recognition*, 29(10):1713–1717, 1996. 81

[25] B. D'andréa-novel, G. Campion, and G. Bastin. Control of wheeled mobile robots not satisfying ideal velocity constraints: A singular perturbation approach. *International Journal of Robust and Nonlinear Control*, 5:243–267, 1995. 2, 3

[26] A. De Luca and M.D. Di Benedetto. Control of nonholonomic systems via dynamic compensation. *Kybernetika*, 29(6):593–608, 1993. 42

[27] A. De Luca, G. Oriolo, and M. Vendittelli. Control of wheeled mobile robots: An experimental overview. *Ramsete*, pages 181–226, 2001. 13, 15

[28] C.C. De Wit, G. Bastin, and B. Siciliano. *Theory of robot control*. Springer-Verlag New York, Inc. Secaucus, NJ, USA, 1996. 3

[29] A. Dib and H. Siguerdidjane. Experimental results of integral sliding mode controller for a nonholonomic mobile robot. In *8th International Conference in Control, Automation and Robotics, Netherlands*, 2011. xxxii, 7

[30] A. Dib and H. Siguerdidjane. Visual servoing for a wheeled mobile robot using image moments: Experimental results. In *Proceedings of the IASTED International Conference on Intelligent Systems and Control, Cambridge, UK*, pages 190–197, 2011. xxxiii, 7

[31] A. Dib, N. Zaidi, and H. Siguerdidjane. Robust control and visual servoing of an UAV. In *Proceedings of 17th IFAC World Congress*, volume 17, pages 5730–5735, 2008. 3

[32] WE Dixon, ZP Jiang, and DM Dawson. Global exponential setpoint control of wheeled mobile robots: a Lyapunov approach* 1. *Automatica*, 36(11):1741–1746, 2000. 2

[33] W. Dong and W. Huo. Tracking control of wheeled mobile robots with unknown dynamics. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 4, pages 2645–2650, 1999. 2

[34] W. Dong and WL Xu. Adaptive tracking control of uncertain nonholonomic dynamic system. *IEEE Transactions on Automatic Control*, 46(3):450–454, 2001. 2

[35] F. Du and M. Brady. Self-calibration of the intrinsic parameters of cameras for active vision systems. In *Computer Vision and Pattern Recognition, 1993. Proceedings CVPR'93., 1993 IEEE Computer Society Conference on*, pages 477–482. IEEE, 1993. 86

[36] F. Fahimi. *Autonomous robots: modeling, path planning, and control*. Springer Verlag, 2008. 19

[37] O. Faugeras. *Three-dimensional computer vision: a geometric viewpoint*. the MIT Press, 1993. 81, 154

[38] R. Fierro and FL Lewis. Control of a nonholonomic mobile robot: backstepping kinematics into dynamics. In *Proceedings of the 34th IEEE Conference on Decision and Control*, volume 4, pages 3805–3810. IEEE, 1995. 3

[39] R. Fierro and FL Lewis. Control of a nonholomic mobile robot: Backstepping kinematics into dynamics. *Journal of Robotic Systems*, 14(3):149–163, 1997. 2

[40] T. Fitzgibbons and E. Nebot. Application of vision in simultaneous localization & mapping. *Intelligent autonomous systems 7*, page 92, 2002. 4

[41] J. Flusser and T. Suk. Pattern recognition by affine moment invariants. *Pattern recognition*, 26(1):167–174, 1993. 80

[42] R.C. Gonzalez and R.E. Woods. *Digital image processing*. Prentice Hall Upper Saddle River, NJ, 2002. 79, 146

[43] J. Guldner and VI Utkin. Stabilization of non-holonomic mobile robots using Lyapunov functions for navigation and sliding mode control. In *Proceedings of the 33rd IEEE Conference on Decision and Control*, volume 3, pages 2967–2972. IEEE, 1994. 45

[44] T. Hamel and R. Mahony. Visual servoing of an under-actuated dynamic rigid-body system: an image-based approach. *Robotics and Automation, IEEE Transactions on*, 18(2):187–198, 2002. 4

[45] R. Hartley. Self-calibration from multiple views with a rotating camera. *Computer Vision—ECCV'94*, pages 471–478, 1994. 86

[46] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge Univ Pr, 2003. 81

[47] J. Hill and W. T. Park. Real time control of a robot with a mobile camera. In *9th International Symposium on Industrial Robot*, page 233–246, Washington, DC, mar 1979. 3

[48] R. Horaud and O. Monga. *Vision par ordinateur: outils fondamentaux*. Hermes, 1995. 86

[49] K. Hosoda and M. Asada. Versatile visual servoing without knowledge of true jacobian. In *Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems*, volume 1, pages 186–193. IEEE, 1994. 94

[50] M.K. Hu. Visual pattern recognition by moment invariants. *Information Theory, IRE Transactions on*, 8(2):179–187, 1962. 4, 80

[51] S. Hutchinson, G.D. Hager, and P.I. Corke. A tutorial on visual servo control. *IEEE transactions on robotics and automation*, 12(5):651–670, 1996. 3, 4

[52] A. Isidori. *Nonlinear Control Systems*. Springer-Verlag, 3rd edition, 1995. xiv, xxii, 13, 60

[53] XY Jiang and H. Bunke. Simple and fast computation of moments. *Pattern Recognition*, 24(8):801–806, 1991. 81

[54] Z.P. Jiang and H. Nijmeijer. Tracking control of mobile robots: a case study in backstepping. *Automatica*, 33(7):1393–1399, 1997. 2, 3

[55] Z.P. Jiang, E. Lefeber, and H. Nijmeijer. Saturated stabilization and tracking of a nonholonomic mobile robot. *Systems and Control Letters*, 42(5):327–332, 2001. 2

[56] I. Kamon, E. Rivlin, and E. Rimon. A new range-sensor based globally convergent navigation algorithm for mobile robots. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 1, pages 429–435. IEEE, 1996. xviii, 34

[57] Y. Kanayama, A. Nilipour, and CA Lelm. A locomotion control method for autonomous vehicles. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1315–1317. IEEE, 1988. 14

[58] Y. Kanayama, Y. Kimura, F. Miyazaki, and T. Noguchi. A stable tracking control scheme for an autonomous mobile robot. proc. of IEEE Int. Conf. on Robotics and Automation. *Proceedings of IEEE International Conference on Robotics and Automation*, pages 384–389, 1990. 2, 3, 67

[59] I. Kolmanovsky and NH McClamroch. Developments in nonholonomic control problems. *IEEE Control Systems Magazine*, 15(6):20–36, 1995. 17

[60] M. Krstic, I. Kanellakopoulos, and P.V. Kokotovic. *Nonlinear and adaptive control design*. John Wiley & Sons New York, 1995. 60

[61] J.C. Latombe. *Robot motion planning*. Springer Verlag, 1990. 3, 70

[62] J.G. Leu. Computing a shape's moments from its boundary. *Pattern Recognition*, 24(10):949–957, 1991. 81

[63] B.C. Li. A new computation of geometric moments. *Pattern Recognition*, 26(1): 109–113, 1993. 81

[64] B.C. Li and J. Shen. Fast computation of moment invariants. *Pattern Recognition*, 24(8):807–813, 1991. 81

[65] Y.H. Liu and S. Arimoto. Path planning using a tangent graph for mobile robots among polygonal and curved obstacles. *The International journal of robotics research*, 11(4):376, 1992. 34

[66] V.J. Lumelsky and A.A. Stepanov. Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. *Algorithmica*, 2(1):403–430, 1987. 33

[67] Y. Mezouar and F. Chaumette. Path planning for robust image-based control. *IEEE Transactions on Robotics and Automation*, 18(4):534–549, 2002. 4

[68] A. Micaelli, C. Samson, and Institut national de recherche en informatique et en automatique (France). *Trajectory Tracking for Unicycle-type and Two-steering Wheels Mobile Robots*. Citeseer, 1993. 3

[69] J. More. The Levenberg-Marquardt algorithm: implementation and theory. *Numerical analysis*, pages 105–116, 1978. 154

[70] R. Mukundan and KR Ramakrishnan. *Moment functions in image analysis: theory and applications*. World Scientific Pub Co Inc, 1998. 4, 80

[71] R.M. Murray. Control of nonholonomic systems using chained form. *Dynamic and control of mechanical systems. The falling cat and related problems*, 1:219–245, 1991. 17

[72] R.M. Murray and S.S. Sastry. Steering nonholonomic systems in chained form. In *Proceedings of the 30th IEEE Conference on Decision and Control*, pages 1121–1126, 1991. 18

[73] J.I. Neimark and N.A. Fufaev. *Dynamics of nonholonomic systems*. Amer Mathematical Society, 1972. 2

[74] W.L. Nelson and I.J. Cox. Local path control for an autonomous vehicle. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1504–1510. IEEE, 1988. 14

[75] H. Nijmeijer and A.J. van der Schaft. *Nonlinear dynamical control systems*. Springer, 1990. ISBN 038797234X. 60

[76] W. Oelen and J. Van Amerongen. Robust tracking control of two-degrees-of-freedom mobile robots. *Control Engineering Practice*, 2(2):333–340, 1994. 3

[77] G. Oriolo, A. De Luca, and M. Vendittelli. WMR control via dynamic feedback linearization: design, implementation, and experimental validation. *IEEE Transactions on Control Systems Technology*, 10(6):835–852, 2002. 3

[78] R. Ortega, A. Loria, P.J. Nicklasson, and H. Sira-Ramirez. *Passivity-based control of Euler-Lagrange systems*, volume 388. Springer Berlin, 1998. 60

[79] A. Papoulis, S.U. Pillai, and S. Unnikrishna. *Probability, random variables, and stochastic processes*. McGraw-Hill New York, 1991. 146

[80] R. Penrose. A generalized inverse for matrices. In *Mathematical proceedings of the Cambridge philosophical society*, volume 51, pages 406–413. Cambridge Univ Press, 1955. 94

[81] W. Philips. A new fast algorithm for moment computation. *Pattern Recognition*, 26(11):1619–1621, 1993. 81

[82] R.J. Prokop and A.P. Reeves. A survey of moment-based techniques for unoccluded object representation and recognition. *CVGIP: Graphical Models and Image Processing*, 54(5):438–460, 1992. 4

[83] P. Puget and T. Skordas. Calibrating a mobile camera. *Image and vision computing*, 8(4):341–348, 1990. 86

[84] R.S. Rao, V. Kumar, and C.J. Taylor. Planning and control of mobile robots in image space from overhead cameras. pages 2185–2190, 2005. 4

[85] C. Samson. Control of chained systems application to path following and time-varying point-stabilization of mobile robots. *IEEE Transactions on Automatic Control*, 40(1):64–77, 1995. 2

[86] C. Samson, K. Ait-Abderrahim, and V. Iria. Feedback control of a nonholonomic wheeled cart in cartesian space. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1136–1141, 1991. 2

[87] S. Saripalli, J.F. Montgomery, and G.S. Sukhatme. Vision-based autonomous landing of an unmanned aerial vehicle. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 3, pages 2799–2804. IEEE, 2002. 4

[88] J.T. Schwartz and M. Sharir. Algorithmic motion planning in robotics. *Handbook of theoretical computer science*, pages 391–430, 1990. 3

[89] R. Sepulchre, M. Janković, and P.V. Kokotović. *Constructive nonlinear control*, volume 9. Springer New York, 1997. 60

[90] O. Shakernia, Y. Ma, T.J. Koo, and S. Sastry. Landing an unmanned air vehicle: vision based motion estimation and nonlinear control. *Asian Journal of Control*, 1(3):128–145, 1999. 4

[91] Y. Shirai and H. Inoue. Guiding a robot by visual feedback in assembling tasks. *Pattern Recognition*, 5:99–108, 1973. 3

[92] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo. *Robotics: modelling, planning and control*. Springer Verlag, 2008. 39, 78, 147

[93] M.H. Singer. A general approach to moment calculation for polygons and line segments. *Pattern Recognition*, 26(7):1019–1028, 1993. 81

[94] J.J. Slotine and SS Sastry. Tracking control of non-linear systems using sliding surfaces, with application to robot manipulators†. *International Journal of Control*, 38(2):465–492, 1983. 44

[95] S. Soatto and P. Perona. Structure-independent visual motion control on the essential manifold. In *In Proceedings of the IFAC Symposium on Robot Control (SYROCO*. Citeseer, 1994. 4

[96] OJ Sordalen and O. Egeland. Exponential stabilization of nonholonomic chained systems. *IEEE Transactions on Automatic Control*, 40(1):35–49, 1995. 2

[97] J. Stewart. *Calculus*. Pacific Grove, CA: Brooks/Cole, 2nd edition, 1991. 145

[98] H.G. Tanner and K.J. Kyriakopoulos. Discontinuous backstepping for stabilization of nonholonomic mobile robots. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 4, pages 3948–3953. Citeseer, 2002. 2

[99] P. Tournassoud. Motion planning for a mobile robot with a kinematic constraint. *Geometry and Robotics*, pages 150–171, 1989. 2

[100] R. Tsai. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. *IEEE Journal of robotics and Automation*, 3(4):323–344, 1987. 86

[101] V. Utkin and J. Shi. Integral sliding mode in systems operating under uncertainty conditions. In *Proceedings of the 35th IEEE on Decision and Control*, volume 4, pages 4591–4596. IEEE, 1996. 45

[102] V.I. Utkin. *Sliding modes in control and optimization*, volume 3. Springer-Verlag Berlin, 1992. 46

[103] G. Walsh, D. Tilbury, S. Sastry, R. Murray, and J.P. Laumond. Stabilization of trajectories for systems with nonholonomic constraints. *IEEE Transactions on Automatic Control*, 39(1):216–222, 1994. 2

[104] G. Xu and Z. Zhang. *Epipolar geometry in stereo, motion, and object recognition: a unified approach*. Springer, 1996. 81, 82

[105] J.M. Yang and J.H. Kim. Sliding mode control for trajectory tracking of nonholonomic wheeled mobile robots. *IEEE Transactions on Robotics and Automation*, 15(3):578–587, 1999. 45

[106] KS Yeung and YP Chen. A new controller design for manipulators using the theory of variable structure systems. *IEEE Transactions on Automatic Control*, 33(2):200–206, 1988. 44

[107] MF Zakaria, LJ Vroomen, PJA Zsombor-Murray, and J. Van Kessel. Fast algorithm for the computation of moment invariants. *Pattern Recognition*, 20(6): 639–643, 1987. 81

[108] E. Zergeroglu, DM Dawson, MS De Queiroz, and S. Nagarkatti. Robust visual-servo control of robot manipulators in the presence of uncertainty. In *Proceedings of the 38th IEEE Conference on Decision and Control*, volume 4, pages 4137–4142. IEEE, 1999. 4

[109] H. Zhang and J.P. Ostrowski. Visual servoing with dynamics: Control of an unmanned blimp. In *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, volume 1, pages 618–623. IEEE, 2002. 4

[110] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000. 86, 88, 151, 152

Abstract: This thesis focuses on the problem of moving and localizing an autonomous mobile robot in its local environments. It is structured into two parts. The first part of the manuscript concerns two basic motion tasks, namely the stabilization and trajectory tracking. Two control strategies were discussed: the integral sliding mode, and the method known as "Immersion and Invariance" for nonlinear control. The second part focuses on both 2D and 3D visual servoing techniques. Image moments were chosen as visual features as they provide a more geometric and intuitive meaning than other features and they are less sensitive to image noise and other measurement errors. A new approach to visual servoing based on image is herein proposed. It is based on the generation of trajectories directly on the image plane (Calculation of the image features corresponding to a given Cartesian path). This approach ensures that the robustness and stability are extended due to the fact that the initial and desired locations of the camera are close. The trajectories obtained guarantee that the target remains in the field of view of the camera and the corresponding movement of the robot is physically feasible. Experimental tests have been conducted, and satisfactory results have been obtained from both implementations regarding motion control and visual servoing strategies. Although developed and tested in the specific context of a unicycle type robot, this work is generic enough to be applied to other types of vehicles.

Résumé: Dans ce travail de thèse, on s'intéresse au problème de déplacement et de la localisation d'un robot mobile autonome dans son environnement local. Il est structuré en deux partie. La première partie du manuscrit porte sur les deux tâches de mouvement de base, c'est-à-dire : la stabilisation et le suivi de trajectoire. Deux stratégies de commande ont été traitées: le mode de glissement intégral et la méthode dite "Immersion et Invariance". La deuxième partie porte sur l'asservissement visuel, les deux techniques 2D et 3D d'asservissement visuel ont été appliquées. Les moments d'image ont été choisis comme indices visuels car ils sont moins sensibles au bruit d'image et autres erreurs de mesure. Une nouvelle approche de l'asservissement visuel qui repose sur l'image est ici proposée. Elle est basée sur la génération de trajectoires sur le plan de l'image directement (calcul des valeurs des primitives d'image correspondantes à une trajectoire cartésienne donnée). Cette approche garantit que la robustesse et la stabilité bien connues de l'asservissement 2D ont été étendues en raison du fait que les emplacements initial et désiré de la caméra sont proches. Les trajectoires obtenues garantissent aussi que la cible reste dans le champ de vue de la caméra et que le mouvement du robot correspondant est physiquement réalisable. Des tests expérimentaux ont été effectués et des résultats satisfaisants ont été obtenus à partir des implémentations des stratégiesde commande et d'asservissement visuel. Bien qu'ils soient développés et expérimentés dans le cadre spécifique d'un robot de type unicycle, ces travaux sont assez génériques pour être appliqués sur d'autres types de véhicules.